(4)

SYSTEM THEORETIC MODELS FOR HIGH DENSITY VLSI STRUCTURES

ONR CONTRACT N00014-83-K-0577

Bradley W. Dickinson and William E. Hopkins, Jr.
Co-principal Investigators
Department of Electrical Engineering
Princeton University
Princeton, New Jersey 08544

Final Technical Report
Prepared by Bradley W. Dickinson

Summary

This research project involved the development of mathematical models for analysis, synthesis, and simulation of large systems of interacting devices. The work was motivated by problems that may become important in high density VLSI chips with characteristic feature sizes less than 1 micron: it is anticipated that interactions of neighboring devices will play an important role in the determination of circuit properties. It is hoped that the combination of high device densities and such local interactions can somehow be exploited to increase circuit speed and to reduce power consumption. To address these issues from the point of view of system theory, research was pursued in the areas of nonlinear and stochastic systems and into neural network models.

Statistical models were developed to characterize various features of the dynamic behavior of interacting systems. Random process models for studying the resulting asynchronous modes of operation were investigated. The local interactions themselves may be modeled as stochastic effects. The resulting behavior has been investigated through the use of various scaling limits, and by a combination of other analytical and simulation techniques. Techniques arising in a variety of disciplines where models of interaction have been formulated and explored were considered and adapted for use. Of particular relevance are random field models of spatial interaction, various results concerning stochastic convergence related to the Central Limit Theorem, and some basic ideas about computational complexity related to analog systems. Research into the relations between state space structure and the input-output function of large systems, using geometric and algebraic methods from nonlinear system theory, was performed. Distributed computation models, in the form of artificial neural networks, have been studied because of the great interest in applications of such systems to a variety of problems in pattern classification and signal processing.

Technical Results

A first area of significant progress developed from an investigation of applications of nonlinear system theory to realizability questions for linear filtering. It has been shown [1] that internally nonlinear systems do not produce a broader class of linear input-output behaviors than internally linear systems. This means, for example, that optimal linear filters cannot be realized by "lumped" systems when the associated covariance functions are not separable; thus it is not possible to exploit nonlinear behavior to obtain optimal linear filters for more general signals than the class to which the well-known Kalman filter may be applied. For nonlinear filters that are described by a Volterra series input-output description, a similar kind of result was obtained [4], and more general results for linear filtering in colored (correlated) noise processes were developed using the theory of reproducing kernel Hilbert spaces (RKHS) [5]. The general importance of these results is that there are fundamental structural limitations imposed on the input-output behavior of any well-behaved finite dimensional nonlinear system.

A second area of work was in the formulation of Markov field models for spatially interacting systems, [2] and [10]. A number of models were proposed, and efficient simulation procedures were developed. Several conclusions were drawn on the basis of the work. First, the computational demands of general Markov field models are extremely severe, and this suggests that either approximation methods will probably play an important role in any case where they are to be applied or massively parallel computers such as the Connection Machine will be needed to handle the demands. With no such parallel computing engine available for experimental work, research relied on modest simulation studies, and special attention was given to studies of possibilities for time-scale decomposition and state aggregation. Intriguing nonlinear phenomena similar to phase transitions in quantum physics models are observed in the behavior of locally interacting systems.

After considerable discussion, including the valuable one at the ONR-sponsored workshop on submicron systems, it appears that physical models for interacting quantum systems have not yet reached the stage where applications of Markov field models for large-scale behavioral modeling are realistic. That is to say, the physical

constraints that are necessary to impose on this general form of empirical model are not yet well enough understood. However, based on the success of stochastic models in applications such as binary (and gray-scale) image modeling, we continue to be optimistic that future work on quantum well devices and systems will lead to some applications for Markov field models.

An attempt to use Markov field models in an application involving VLSI systems was made: the phenomenon of *pattern sensitive faults* in dynamic memory devices was considered [10]. However, the results of this preliminary analysis are not very encouraging. Several comments about this particular problem can be made. First, conventional work on testing of memory chips involves the design of a sufficiently rich class of test input sequences. The use of statistical modeling, and system identification/parameter estimation methods for determining faulty chips on the basis of random test input sequences, is a different approach to testing which is not (yet) widely accepted, although statistical models for phenomena affecting component lifetimes (e.g. electromigration of metal at contacts) are common in reliability modeling. Finally, we are unaware of any nonproprietary data on real chips that could be used in evaluating the stochastic modeling approach to testing in a realistic setting. This is crucial for such applications where it is desired to detect rare events (failures) with low error probabilities. The highly developed detection systems used in radar and sonar applications have evolved from an extensive amount of empirical and analytical work. It should be expected that applications of statistical models in VLSI testing will require a similar combination of effort.

The general work done in the area of interacting systems of simple elements suggested that various distributed computational models may be useful in signal processing applications. The same conclusion has been reached by many other researchers starting from a variety of perspectives. To examine this kind of question in a specific setting, we undertook some research to explore the structure of artificial neural network models with an eye towards isolating one or more neural network models that could be implemented with a spatially interacting structure of the type we imagine might be relevant for future generation high density VLSI chips.

The first major accomplishment was a rederivation of the capacity results for

Hopfield associative memory networks [9], [10]. This problem has attracted considerable attention, partly because of its implications that, in one particular but important sense, the number of stored memories that can be achieved on average is disappointingly small (growing sublinearly with the size of the network). A new analysis was carried out, using probabilistic methods related to the Central Limit Theorem for dependent, *exchangeable*, random variables. This offers an advantage over the earlier combinatorial approach because it gives a way of exhibiting how constraints on interconnections and how stochastic neuron models affect asymptotic capacity. Some further analysis, giving bounds for the number of spurious memories, was also carried out [11].

The second major thrust in the neural networks area involved a family of structured, locally interconnected networks based on trellis graphs associated with linear finite-state systems. Systems of this kind are used to generate convolutional codes for digital communications. It was found that the combinatorial optimization problem of finding a shortest part through a segment of a trellis graph could be solved by a suitably formulated (Grossberg-type) neural network [7], [8], [11]. This provides a localized, distributed solution to the shortest path problem quite different in spirit than the dynamic programming solution (Viterbi algorithm) which is widely used in practice. Furthermore, this use of neural networks seems to offer a particularly nice method for representing data with distributed redundancy and natural capabilities for efficient fault tolerant implementation.

Finally, there was some research work that built upon on-going work concerning analog computation. In particular, the work described in [6] describes a framework for analysis of the complexity of physical systems, e.g. electronic circuits, neural networks, etc., based on the recognized (or at least widely believed) computational intractability of certain combinatorial optimization problems, the class of NP-complete problems. Since neural networks and more general nonlinear circuits have been proposed as a means of solving NP-complete problems like the "Traveling Salesman Problem," (by Hopfield and Tank, Chua, and others), the results of [6], namely that the scaling difficulties encountered by such solutions as the problem size increases, are quite naturally to be expected on the basis of complexity theory

arguments.

Participants

Professors Bradley W. Dickinson and William E. Hopkins, Jr. were the principal investigators for the project. Several graduate students provided major contributions; Carla Schwartz, Anthony Kuh, and Thomas Petsche made significant contributions as indicated in the list of publications.

Publication List

[1] Characterizing Finite Dimensional Filters for the Linear Innovations of Continuous Time Random Processes, C.A. Schwartz and B.W. Dickinson; *IEEE Trans. Automatic Control*, **30**, 1985, 312-315. (Also presented at the 1984 Conf. on Decision and Control.)

[2] Stochastic Models of Interacting Systems, T. Kuh and B.W. Dickinson; *Proceeding of the 23rd Conference on Decision and Control,* Las Vegas, NV, 1984, 1488-1489.

[3] Stochastic Systems with Local Interactions, B.W. Dickinson and W.E. Hopkins, Jr.; *Proceedings of the 1985 International Symposium on Circuits and Systems,* Kyoto, Japan, 1985, 897.

[4] A Finite Dimensional Realization Theorem for Discrete-Time Nonlinear Systems, C.A. Schwartz and B.W. Dickinson; in *Theory and Applications on Nonlinear Control*, (ed. by C.I. Byrnes and A. Lindquist), North Holland, Amsterdam, 1986, pp. 223-230. (This volume is the refereed proceedings of the 1985 Symposium on Mathematical Theory of Networks and Systems.)

[5] Systems in Reproducing Kernel Hilbert Space: Causality, Realizability, and Separability, C.A. Schwartz and B.W. Dickinson; *IMA J. Mathematical Control and Information*, **3**, 1986, 223-236.

[6] The Complexity of Analog Computation, A. Vergis, K. Steiglitz, and B. Dickinson; *Mathematics and Computers in Simulation,* **28**, 1986, 91-113. A copy of the Abstract and Introduction is included in the Appendix.

[7] A Locally Interconnected Neural Network for Deconvolution, T. Petsche and B.W. Dickinson; *Proceeding of the 21st Conference on Information Sciences and Systems*, Baltimore, MD, 1987, 381-386. A copy is included in the Appendix.

[8] A Trellis-Structured Neural Network, T. Petsche and B.W. Dickinson; in *Neural Information Processing Systems*, (ed. by D.A. Anderson), American Institute of Physics, 1988, 592-601. (This volume is the proceedings of the 1986 NIPS Conference.) A copy is included in the Appendix.

[9] Information Capacity of Associative Memories, A. Kuh and B.W. Dickinson; *IEEE Trans. Information Theory*, **35**, 1989, 59-68. (This work was presented, in part, at the 1986 IEEE International Symposium on Information Theory.) A copy is included in the Appendix.

[10] *Stochastic Models for Interacting Systems*, Anthony Kuh; Ph.D. thesis, Dept. of Electrical Engineering, Princeton University, January 1987. Dr. Kuh is now with the Dept. of Electrical Engineering, Univ. of Hawaii. A copy of the thesis abstract is included in the Appendix.

[11] *Topics in Neural Networks*, Thomas Petsche; Ph.D. theses, Dept. of Electrical Engineering, Princeton University, June 1988. Dr. Petsche is now with Siemens Corporate Research and Support, Inc. A copy of the thesis abstract is included in the Appendix.

Appendix: Copies of Selected Publications

Refer to items [6]-[11] in the Publication List

Mathematics and Computers in Simulation 28 (1986) 91-113
North-Holland

THE COMPLEXITY OF ANALOG COMPUTATION *

Anastasios VERGIS

Department of Computer Science, University of Minnesota, Minneapolis, MN 55455, U.S.A.

Kenneth STEIGLITZ

Department of Computer Science, Princeton University, Princeton, NJ 08544, U.S.A.

Bradley DICKINSON

Department of Electrical Engineering, Princeton University, Princeton, NJ 08544, U.S.A.

We ask if analog computers can solve NP-complete problems efficiently. Regarding this as unlikely, we formulate a strong version of Church's Thesis: that any analog computer can be simulated *efficiently* (in polynomial time) by a digital computer. From this assumption and the assumption that P ≠ NP we can draw conclusions about the operation of physical devices used for computation.

An NP-complete problem, 3-SAT, is reduced to the problem of checking whether a feasible point is a local optimum of an optimization problem. A mechanical device is proposed for the solution of this problem. It encodes variables as shaft angles and uses gears and smooth cams. If we grant Strong Church's Thesis, that P ≠ NP, and a certain "Downhill Principle" governing the physical behavior of the machine, we conclude that it cannot operate successfully while using only polynomial resources.

We next prove Strong Church's Thesis for a class of analog computers described by well-behaved ordinary differential equations, which we can take as representing part of classical mechanics.

We conclude with a comment on the recently discovered connection between spin glasses and combinatorial optimization.

1. Introduction

Analog devices have been used, over the years, to solve a variety of problems. Perhaps most widely known is the Differential Analyzer [4,26], which has been used to solve differential equations. To mention some other examples, in [25] an electronic analog computer is proposed to implement the gradient projection method for linear programming. In [18] the problem of finding a minimum-length interconnection network between given points in the plane is solved with movable and fixed pegs interconnected by strings; a locally optimal solution is obtained by pulling the strings. Another method is proposed there for this problem, based on the fact that soap films form minimal-tension surfaces. Many other examples can be found in books such as [14] and [16], including electrical and mechanical machines for solving simultaneous linear equations and differential equations.

* This work was supported in part by ONR Grants N00014-83-K-0275 and N00014-83-K-0577, NSF Grant ECS-8120037, U.S. Army Research-Durham Grant DAAG29-82-K-0095, and DARPA Contract N00014-82-K-0549.

Given the large body of work on the complexity of Turing-machine computation, and the recent interest in the physical foundations of computation, it seems natural to study the complexity of analog computation. This paper pursues the following line of reasoning: it is generally regarded as likely that $P \neq NP$—that certain combinatorial problems cannot be solved efficiently by digital computers. (Here we use the term *efficient* to mean that the time used by an 'ideal' digital computer is bounded by a polynomial function of the size of the task description. See [9] for discussion of this criterion.) We may ask if such problems can be solved efficiently by other means, in particular, by machines of a nature different from digital computers. We thus come to ask if NP-complete problems can be solved efficiently by physical devices that do not use binary encoding (or, more generally, encoding with any fixed radix). We lump such devices together under the term *analog computer*; in what follows we will use the term *analog computer* to mean any deterministic physical device that uses a fixed number of physical variables to represent each problem variable. This description is admittedly vague and certainly non-mathematical—we mean it to capture the intuitive notion of a 'non-digital' computer. (More about this in the next section.)

We want to emphasize that the question of whether an analog computer can solve an NP-complete problem 'efficiently' is a question about the physical world, while the $P = NP$ question is a mathematical one. However, mathematical models of various kinds provide a formalism that is apparently indispensable for the understanding of physical phenomena. An important connection between the mathematical world of computation and the physical world of computing hardware was discussed by Church. In his 1936 paper [6] he equated the intuitive notion of effective calculability with the two equivalent mathematical characterizations of λ-definability and recursivity. Turing [28] then showed that this notion is equivalent to computability by what we have come to call a Turing machine, so that the intuitive notion of effective calculability is now characterized mathematically by 'Turing-Computability'. This is generally referred to as 'Church's Thesis', or the 'Church–Turing Thesis'. In our context we express this as follows:

> Church's Thesis (CT): Any analog computer with finite resources can be simulated by a digital computer.

What we will come to demand is more than that: we are interested in efficient computation, computation that does not use up resources that grow exponentially with the size of the problem. This requirement leads us to formulate what we call

> Strong Church's Thesis (SCT): Any finite analog computer can be simulated *efficiently* by a digital computer, in the sense that the time required by the digital computer to simulate the analog computer is bounded by a polynomial function of the resources used by the analog computer.

Evidently we will need to give a characterization of analog computers and the resources that they use. This is discussed in the next section. Following that, we argue that certain numerical problems are inherently difficult (i.e. not polynomial) for analog computers, even though they are easy for digital computers.

Something like our Strong Church's Thesis was discussed recently by Feynman [8] in connection with the problem of building a (digital) computer that simulates physics. He says:

> "The rule of simulation that I would like to have is that the number of computer elements required to simulate a large physical system is only to be proportional to the space-time volume of the physical system. I don't want to have an explosion."

We would argue that 'proportional to' be replaced by 'bounded by a polynomial function of', in the spirit of modern computational complexity theory.

A class of mechanical devices is proposed in Section 5. Machines in this class can be used to find local optima for mathematical programming problems. We formalize the physical operation of these machines by a certain 'Downhill Principle'. Basically, it states that if, in our class of mechanical devices, there are feasible 'downhill' directions, the state vector describing the physical system moves in such a direction. We also discuss measuring the resources required by these machines.

In Section 6 we reduce 3-SAT (the problem of whether a Boolean expression in 3-conjunctive normal form has a satisfying truth assignment), to the problem of checking whether a given feasible point is a local optimum of a certain mathematical programming problem. This shows that merely checking for local optimality is NP-hard.

In Section 7 a mechanical device in the class mentioned above is proposed for the solution of 3-SAT. Naturally, the efficient operation of this machine is highly suspect. Be careful to notice that the operation of any machine in practice is a *physics* question, not a question susceptible of ultimate mathematical demonstration. Our analysis must necessarily be based on an idealized mathematical model for the machine. However, we can take the likelihood of $P \neq NP$, plus the likelihood of Strong Church's Thesis, as evidence that in fact such a machine cannot operate with polynomially bounded resources, whatever the particular laws of physics happen to be.

The paradigm that emerges from this line of reasoning is then the following:

> If a strongly NP-complete problem can be solved by an analog computer, and if $P \neq NP$, and if Strong Church's Thesis is true then the analog computer cannot operate successfully with polynomial resources.

We will then prove a restricted form of Strong Church's Thesis, for analog computers governed by well-behaved differential equations. This suggests that any interesting analog computer should rely on some strongly nonlinear behavior, perhaps arising from quantum mechanical mechanisms; however, the problem of establishing Strong Church's Thesis (or even the Weak Thesis) in the case of quantum-mechanical or probabilistic laws is an open problem.

2. Some terminology

We know what a digital computer is; Turing has laid out a model for what a well-defined digital computation must be: it uses a finite set of symbols (without loss of generality $\{0,1\}$) to store information, it can be in only one of a finite set of states, and it operates by a finite set of rules for moving from state to state. Its memory tape is not bounded in length a priori, but only a finite amount of tape can be used for any one computation. What is fundamental about the idea of a Turing Machine and digital computation in general, is that there is a perfect correspondence between the mathematical model and what happens in a reasonable working machine. Being definitely in one of two states is easily arranged in practice, and the operation of real digital computers can be (and usually is) made very reliable.

Neural Information
Processing Systems, D.A.
Anderson, ed., pp. 592-601,
American Institute of
Physics, 1988.

A Trellis-Structured Neural Network*

Thomas Petsche† and Bradley W. Dickinson
Princeton University, Department of Electrical Engineering
Princeton, NJ 08544

Abstract

We have developed a neural network which consists of cooperatively interconnected Grossberg on-center off-surround subnets and which can be used to optimize a function related to the log likelihood function for decoding convolutional codes or more general FIR signal deconvolution problems. Connections in the network are confined to neighboring subnets, and it is representative of the types of networks which lend themselves to VLSI implementation. Analytical and experimental results for convergence and stability of the network have been found. The structure of the network can be used for distributed representation of data items while allowing for fault tolerance and replacement of faulty units.

1 Introduction

In order to study the behavior of locally interconnected networks, we have focused on a class of "trellis-structured" networks which are similar in structure to multilayer networks [5] but use symmetric connections and allow every neuron to be an output. We are studying such locally interconnected neural networks because they have the potential to be of great practical interest. Globally interconnected networks, e.g., Hopfield networks [3], are difficult to implement in VLSI because they require many long wires. Locally connected networks, however, can be designed to use fewer and shorter wires.

In this paper, we will describe a subclass of trellis-structured networks which optimize a function that, near the global minimum, has the form of the log likelihood function for decoding convolutional codes or more general finite impulse response signals. Convolutional codes, defined in section 2, provide an alternative representation scheme which can avoid the need for global connections. Our network, described in section 3, can perform maximum likelihood sequence estimation of convolutional coded sequences in the presence of noise. The performance of the system is optimal for low error rates.

The specific application for this network was inspired by a signal decomposition network described by Hopfield and Tank [6]. However, in our network, there is an emphasis on local interconnections and a more complex neural model, the Grossberg on-center off-surround network [2], is used. A modified form of the Gorssberg model is described in section 4. Section 5 presents the main theoretical results of this paper. Although the deconvolution network is simply a set of cooperatively interconnected

*Supported by the Office of Naval Research through grant N00014-83-K-0577 and by the National Science Foundation through grant ECS84-05460.

†Permanent address: Siemens Corporate Research and Support, Inc., 105 College Road East, Princeton, NJ 08540.

on-center off-surround subnetworks, and absolute stability for the individual subnetworks has been proven [1], the cooperative interconnections between these subnets make a similar proof difficult and unlikely. We have been able, however, to prove equiasymptotic stability in the Lyapunov sense for this network given that the gain of the nonlinearity in each neuron is large. Section 6 will describe simulations of the network that were done to confirm the stability results.

2 Convolutional Codes and MLSE

In an error correcting code, an input sequence is transformed from a b-dimensional input space to an M-dimensional output space, where $M \geq b$ for error correction and/or detection. In general, for the b-bit input vector $\mathbf{U} = (u_1, \ldots, u_b)$ and the M-bit output vector $\mathbf{V} = (v_1, \ldots, v_M)$, we can write $\mathbf{V} = F(u_1, \ldots, u_b)$. A convolutional code, however, is designed so that relatively short subsequences of the input vector are used to determine subsequences of the output vector. For example, for a rate 1/3 convolutional code (where $M \approx 3b$), with input subsequences of length 3, we can write the output, $\mathbf{V} = (\mathbf{v}_1, \ldots, \mathbf{v}_b)$ for $\mathbf{v}_i = (v_{i,1}, v_{i,2}, v_{i,3})$, of the encoder as a convolution of the input vector $\mathbf{U} = (u_1, \ldots, u_b, 0, 0)$ and three generator sequences

$$\mathbf{g}_0 = (1\ 1\ 1) \qquad \mathbf{g}_1 = (1\ 1\ 0) \qquad \mathbf{g}_2 = (0\ 1\ 1).$$

This convolution can be written, using modulo-2 addition, as

$$\mathbf{v}_i = \sum_{k=\max(1,i-2)}^{i} u_k \mathbf{g}_{i-k} \tag{1}$$

In this example, each 3-bit output subsequence, $\mathbf{v}_i$, of $\mathbf{V}$ depends only on three bits of the input vector , i.e., $\mathbf{v}_i = f(u_{i-2}, u_{i-1}, u_i)$. In general, for a rate $1/n$ code, the *constraint length*, K, is the number of bits of the input vector that uniquely determine each n-bit output subsequence. In the absence of noise, any subsequences in the input vector separated by more than K bits (i.e., that do not overlap) will produce subsequences in the output vector that are independent of each other.

If we view a convolutional code as a special case of block coding, this rate 1/3, $K = 3$ code converts a b-bit input word into a codeword of length $3(b + 2)$ where the 2 is added by introducing two zeros at the end of every input to "zero-out" the code. Equivalently, the coder can be viewed as embedding 2^b memories into a $2^{3(b+2)}$-dimensional space. The minimum distance between valid memories or codewords in this space is the *free distance* of the code, which in this example is 7. This implies that the code is able to correct a minimum of three errors in the received signal.

For a convolutional code with constraint length K, the encoder can be viewed as a finite state machine whose state at time i is determined by the $K - 1$ input bits, $u_{i-k}, \ldots, u_{i-1}$. The encoder can also be represented as a trellis graph such as the one shown in figure 1 for a $K = 3$, rate 1/3 code. In this example, since the constraint length is three, the two bits u_{i-2} and u_{i-1} determine which of four possible states the encoder is in at time i. In the trellis graph, there is a set of four nodes arranged in a vertical column, which we call a stage, for each time step i. Each node is labeled with the associated values of u_{i-2} and u_{i-1}. In general, for a rate $1/n$ code, each stage of the trellis graph contains 2^{K-1} nodes, representing an equal number of possible states. A trellis graph which contains S stages therefore fully describes the operation of the encoder for time steps 1 through S. The graph is read from left to right and the upper edge leaving the right side of a node in stage i is followed if u_i is a zero; the lower edge

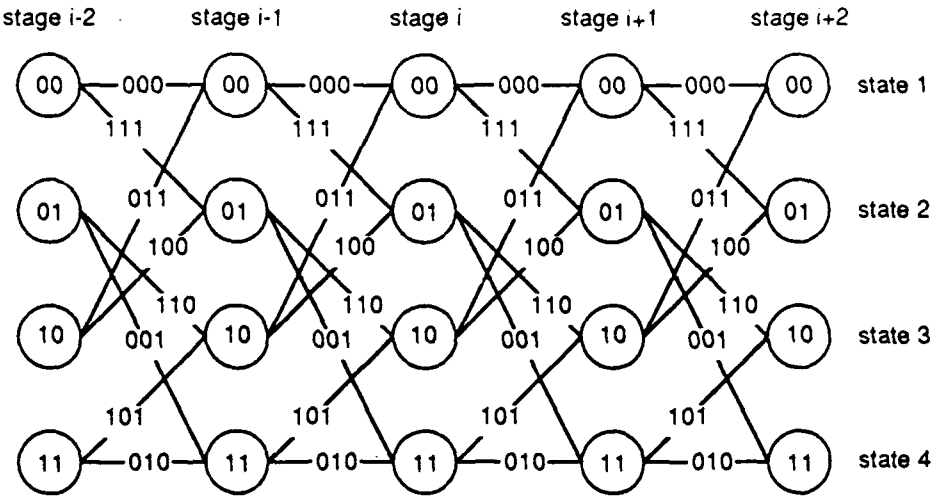

Figure 1: Part of the trellis-code representation for a rate 1/3, $K = 3$ convolutional code.

if u_i is a one. The label on the edge determined by u_i is $\mathbf{v}_i$, the output of the encoder given by equation 1 for the subsequence u_{i-2}, u_{i-1}, u_i.

Decoding a noisy sequence that is the output of a convolutional coder plus noise is typically done using a maximum likelihood sequence estimation (MLSE) decoder which is designed to accept as input a possibly noisy convolutional coded sequence, $\mathbf{R}$, and produce as output the maximum likelihood estimate, $\hat{\mathbf{V}}$, of the original sequence, $\mathbf{V}$. If the set of possible $n(b+2)$-bit encoder output vectors is $\{\mathbf{X}_m : m = 1, ..., 2^{n(b+2)}\}$ and $\mathbf{x}_{m,i}$ is the ith n-bit subsequence of $\mathbf{X}_m$ and $\mathbf{r}_i$ is the ith n-bit subsequence of $\mathbf{R}$ then

$$\hat{\mathbf{V}} = \arg\max_{\mathbf{X}_m} \prod_{i=1}^{b} \mathrm{P}(\mathbf{r}_i \mid \mathbf{x}_{m,i}) \tag{2}$$

That is, the decoder chooses the $\mathbf{X}_m$ that maximizes the conditional probability, given $\mathbf{X}_m$, of the received sequence.

A binary symmetric channel (BSC) is an often used transmission channel model in which the decoder produces output sequences formed from an alphabet containing two symbols and it is assumed that the probability of either of the symbols being affected by noise so that the other symbol is received is the same for both symbols. In the case of a BSC, the log of the conditional probability, $\mathrm{P}(\mathbf{r}_i \mid \mathbf{x}_{m,i})$, is a linear function of the Hamming distance between $\mathbf{r}_i$ and $\mathbf{x}_{m,i}$ so that maximizing the right side of equation 2 is equivalent to choosing the $\mathbf{X}_m$ that has the most bits in common with $\mathbf{R}$. Therefore, equation 2 can be rewritten as

$$\hat{\mathbf{V}} = \arg\max_{\mathbf{X}_m} \sum_{i=1}^{b} \sum_{l=1}^{n} I_{r_{i,l}}(x_{m,i,l}) \tag{3}$$

where $\mathbf{x}_{m,i,l}$ is the lth bit of the ith subsequence of $\mathbf{X}_m$ and $I_a(b)$ is the indicator function: $I_a(b) = 1$ if and only if a equals b.

For the general case, maximum likelihood sequence estimation is very expensive since the number of possible input sequences is exponential in b. The Viterbi algorithm [7], fortunately, is able to take advantage of the structure of convolutional codes and their trellis graph representations to reduce the complexity of the decoder so that

it is only exponential in K (in general $K \ll b$). An optimum version of the Viterbi algorithm examines all b stages in the trellis graph, but a more practical and very nearly optimum version typically examines approximately $5K$ stages, beginning at stage i, before making a decision about u_i.

3 A Network for MLSE Decoding

The structure of the network that we have defined strongly reflects the structure of a trellis graph. The network usually consists of $5K$ subnetworks, each containing 2^{K-1} neurons. Each subnetwork corresponds to a stage in the trellis graph and each neuron to a state. Each stage is implemented as an "on-center off-surround" competitive network [2], described in more detail in the next section, which produces as output a contrast enhanced version of the input. This contrast enhancement creates a "winner take all" situation in which, under normal circumstances, only one neuron in each stage —the neuron receiving the input with greatest magnitude — will be on. The activation pattern of the network after it reaches equilibrium indicates the decoded sequence as a sequence of "on" neurons in the network. If the j-th neuron in subnet i, $\mathcal{N}_{i,j}$ is on, then the node representing state j in stage i lies on the network's estimate of the most likely path.

For a rate $1/n$ code, there is a symmetric cooperative connection between neurons $\mathcal{N}_{i,j}$ and $\mathcal{N}_{i+1,k}$ if there is an edge between the corresponding nodes in the trellis graph. If $(x_{i,j,k,1}, \ldots, x_{i,j,k,n})$ are the encoder output bits for the transition between these two nodes and $(r_{i,1}, \ldots, r_{i,n})$ are the received bits, then the connection weight for the symmetric cooperative connection between $\mathcal{N}_{i,j}$ and $\mathcal{N}_{i+1,k}$ is

$$m_{i,j,k} = \frac{1}{n} \sum_{l=1}^{n} I_{r_{i,l}}(x_{i,j,k,l}) \tag{4}$$

If there is no edge between the nodes, then $m_{i,j,k} = 0$.

Intuitively, it is easiest to understand the action of the entire network by examining one stage. Consider the nodes in stage i of the trellis graph and assume that the conditional probabilities of the nodes in stages $i-1$ and $i+1$ are known. (All probabilities are conditional on the received sequence.) Then the conditional probability of each node in stage i is simply the sum of the probabilities of each node in stages $i-1$ and $i+1$ weighted by the conditional transition probabilities. If we look at stage i in the network, and let the outputs of the neighboring stages $i-1$ and $i+1$ be fixed with the output of each neuron corresponding to the "likelihood" of the corresponding state at that stage, then the final outputs of the neurons $\mathcal{N}_{i,j}$ will correspond to the "likelihood" of each of the corresponding states. At equilibrium, the neuron corresponding to the most likely state will have the largest output.

4 The Neural Model

The "on-center off-surround" network[2] is used to model each stage in our network. This model allows the output of each neuron to take on a range of values, in this case between zero and one, and is designed to support contrast enhancement and competition between neurons. The model also guarantees that the final output of each neuron is a function of the relative intensity of its input as a fraction of the total input provided to the network.

Using the "on-center off-surround" model for each stage and the interconnection weights, $m_{i,j,k}$, defined in equation 4, the differential equation that governs the instantaneous activity of the neurons in our deconvolution network with S stages and N states in each stage can be written as

$$\begin{aligned}\dot{u}_{i,j} = -Au_{i,j} + (B - u_{i,j})\Big(f(u_{i,j}) + \sum_{k=1}^{N}[m_{i-1,k,j}f(u_{i-1,k}) + m_{i,j,k}f(u_{i+1,k})]\Big) \\ - (C + u_{i,j})\sum_{k\neq j}^{N}\Big(f(u_{i,k}) + \sum_{l=1}^{N}[m_{i-1,k,l}f(u_{i-1,k}) + m_{i,l,k}f(u_{i+1,k})]\Big)\end{aligned} \tag{5}$$

where $f(x) = (1 + e^{-\lambda x})^{-1}$, λ is the gain of the nonlinearity, and A, B, and C are constants

For the analysis to be presented in section 5, we note that equation 5 can be rewritten more compactly in a notation that is similar to the equation for additive analog neurons given in [4]:

$$\dot{u}_{i,j} = -Au_{i,j} - \sum_{k=1}^{S}\sum_{l=1}^{N}(u_{i,j}S_{i,j,k,l}f(u_{k,l}) - T_{i,j,k,l}f(u_{k,l})) \tag{6}$$

where, for $1 \leq l \leq N$,

$$\begin{aligned} S_{i,j,i,l} &= 1 & T_{i,j,i,j} &= B \\ S_{i,j,i-1,l} &= \sum_q m_{i-1,l,q} & T_{i,j,i,l} &= -C \quad \forall\, l \neq j \\ S_{i,j,i+1,l} &= \sum_q m_{i,q,l} & T_{i,j,i-1,l} &= Bm_{i-1,l,j} - C\sum_{q\neq j} m_{i-1,l,q} \\ S_{i,j,k,l} &= 0 \quad \forall\, k \notin \{i-1, i, i+1\} & T_{i,j,i+1,l} &= Bm_{i,j,l} - C\sum_{q\neq j} m_{i,q,l} \end{aligned} \tag{7}$$

To eliminate the need for global interconnections within a stage, we can add two summing elements to calculate

$$X_i = \sum_{j=1}^{N} f(x_{i,j}) \quad \text{and} \quad J_i = \sum_{j=1}^{N}\sum_{k=1}^{N}[m_{i-1,k,j}f(u_{i-1,k}) + m_{i,j,k}f(u_{i+1,k})] \tag{8}$$

Using these two sums allows us to rewrite equation 5 as

$$\dot{u}_{i,j} = -Au_{i,j} + (B + C)(f(u_{i,j}) + I_{i,j}) - u_{i,j}(X_i + J_i) \tag{9}$$

This form provides a more compact design for the network that is particularly suited to implementation as a digital filter or for use in simulations since it greatly reduces the calculations required.

5 Stability of the Network

The end of section 3 described the desired operation of a single stage, given that the outputs of the neighboring stages are fixed. It is possible to show that in this situation a single stage is stable. To do this, fix $f(u_{k,l})$ for $k \in \{i-1, i+1\}$ so that equation 6 can be written in the form originally proposed by Grossberg [2]:

$$\dot{u}_{i,j} = -Au_{i,j} + (B - u_{i,j})(I_{i,j} + f(u_{i,j})) - (u_{i,j} + C)\Big(\sum_{k=1}^{N} I_{i,k} + \sum_{k=1}^{N} f(u_{i,k})\Big) \tag{10}$$

where $I_{i,j} = \sum_{k=1}^{N} [m_{i-1,k,j} f(u_{i-1,k}) + m_{i,j,k} f(u_{i+1,k})]$.

Equation 10 is a special case of the more general nonlinear system

$$\dot{x}_i = a_i(x_i)\Big(b_i(x_i) - \sum_{k=1}^{n} c_{i,k} d_k(x_k)\Big) \tag{11}$$

where: (1) $a_i(x_i)$ is continuous and $a_i(x_i) > 0$ for $x_i \geq 0$; (2) $b_i(x_i)$ is continuous for $x_i \geq 0$; (3) $c_{i,k} = c_{k,i}$; and (4) $d_i(x_i) \geq 0$ for all $x_i \in (-\infty, \infty)$. Cohen and Grossberg [1] showed that such a system has a global Lyapunov function:

$$V(\mathbf{x}) = -\sum_{i=1}^{n} \int_0^{x_i} b_i(\xi_i) d_i'(\xi_i) d(\xi_i) + \frac{1}{2} \sum_{j=1}^{n} \sum_{k=1}^{n} c_{j,k} d_j(x_j) d_k(x_k) \tag{12}$$

and that, therefore, such a system is equiasymptotically stable for all constants and functions satisfying the four constraints above. In our case, this means that a single stage has the desired behavior when the neighboring stages are fixed. If we take the output of each neuron to correspond to the likelihood of the corresponding state then, if the two neighboring stages are fixed, stage i will converge to an equilibrium point where the neuron receiving the largest input will be on and the others will be off, just as it should according to section 2.

It does not seem possible to use the Cohen-Grossberg stability proof for the entire system in equation 5. In fact, Cohen and Grossberg note that networks which allow cooperative interactions define systems for which no stability proof exists [1].

Since an exact stability proof seems unlikely, we have instead shown that in the limit as the gain, λ, of the nonlinearity gets large the system is asymptotically stable. Using the notation in [4], define $V_i = f(u_i)$ and a normalized nonlinearity $\bar{f}(\cdot)$ such that $\bar{f}^{-1}(V_i) = \lambda u_i$. Then we can define an energy function for the deconvolution network to be

$$E = -\frac{1}{2} \sum_{i,j,k,l} T_{i,j,k,l} V_{i,j} V_{k,l} - \sum_{i,j} \frac{1}{\lambda} \Big(-A - \sum_{k,l} S_{i,j,k,l} V_{k,l}\Big) \int_{\frac{1}{2}}^{V_{k,l}} \bar{f}^{-1}(\zeta)\, d\zeta \tag{13}$$

The time derivative of E is

$$\begin{aligned} \dot{E} = -\sum_{i,j} \frac{dV_{i,j}}{dt} \Big(& -A u_{i,j} - u_{i,j} \sum_{k,l} S_{i,j,k,l} V_{k,l} + \sum_{k,l} T_{i,j,k,l} V_{k,l} \\ & - \frac{1}{\lambda} \sum_{k,l} S_{i,j,k,l} \int_{\frac{1}{2}}^{V_{k,l}} \bar{f}^{-1}(\zeta) d\zeta \Big) \end{aligned} \tag{14}$$

It is difficult to prove that $\dot{E}$ is nonpositive because of the last term in the parentheses. However, for large gain, this term can be shown to have a negligible effect on the derivative.

It can be shown that for $f(u) = (1 + e^{-\lambda u})^{-1}$, $\int_{\frac{1}{2}}^{V_i} \bar{f}_i^{-1}(\zeta)\, d\zeta$ is bounded above by log(2). In this deconvolution network, there are no connections between neurons unless they are in the same or neighboring stages, i.e., $S_{i,j,k,l} = 0$ for $|i - k| > 1$ and l is restricted so that $0 \leq l \leq S$, so there are no more than $3S$ non-zero terms in the problematical summation. Therefore, we can write that

$$\lim_{\lambda \to \infty} -\frac{1}{\lambda} \sum_{k,l} S_{i,j,k,l} \int_{\frac{1}{2}}^{V_{k,l}} \bar{f}^{-1}(\zeta)\, d\zeta = 0$$

Then, in the limit as $\lambda \rightarrow \infty$, the terms in parentheses in equation 14 converge to $\dot{u}_i$ in equation 6, so that $\lim_{\lambda \rightarrow \infty} \dot{E} = \sum_{i,j} \frac{dV_{i,j}}{dt} \dot{u}_i$. Using the chain rule, we can rewrite this as

$$\lim_{\lambda \rightarrow \infty} \dot{E} = -\sum_{i,j} \left(\frac{dV_{i,j}}{dt} \right)^2 \left(\frac{d}{dV_{i,j}} f^{-1}(V_{i,j}) \right)$$

It can also be shown that that, if $f(\cdot)$ is a monotonically increasing function then $\frac{d}{dV_i} f^{-1}(V_i) > 0$ for all V_i. This implies that for all $\mathbf{u} = (u_{i,1}, \ldots, u_{N,S})$, $\lim_{\lambda \rightarrow \infty} \dot{E} \leq 0$, and, therefore, for large gains, E as defined in equation 13 is a Lyapunov function for the system described by equation 5 and the network is equiasymtotically stable.

If we apply a similar asymptotic argument to the energy function, equation 13 reduces to

$$E = -\frac{1}{2} \sum_{i,j,k,l} T_{i,j,k,l} V_{i,j} V_{k,l} \tag{15}$$

which is the Lyapunov function for a network of discontinuous on-off neurons with interconnection matrix $\mathbf{T}$. For the binary neuron case, it is fairly straight forward to show that the energy function has minima at the desired decoder outputs if we assume that only one neuron in each stage may be on and that B and C are appropriately chosen to favor this. However, since there are $O(S^2N)$ terms in the disturbance summation in equation 15, convergence in this case is not as fast as for the derivative of the energy function in equation 13, which has only $O(S)$ terms in the summation.

6 Simulation Results

The simulations presented in this section are for the rate $1/3$, $K = 3$ convolutional code illustrated in figure 1. Since this code has a constraint length of 3, there are 4 possible states in each stage and an MLSE decoder would normally examine a minimum of $5K$ subsequences before making a decision, we will use a total of 16 stages. In these simulations, the first and last stage are fixed since we assume that we have prior knowledge or a decision about the first stage and zero knowledge about the last stage. The transmitted codeword is assumed to be all zeros.

The simulation program reads the received sequence from standard input and uses it to define the interconnection matrix W according to equation 4. A relaxation subroutine is then called to simulate the performance of the network according to an Euler discretization of equation 5. Unit time is then defined as one RC time constant of the unforced system. All variables were defined to be single precision (32 bit) floating point numbers.

Figure 2a shows the evolution of the network over two unit time intervals with the sampling time $T = 0.02$ when the received codeword contains no noise. To interpret the figure, recall that there are 16 stages of 4 neurons each. The output of each stage is a vertical set of 4 curves. The upper-left set is the output of the first stage; the upper-most curve is the output of the first neuron in the stage. For the first stage, the first neuron has a fixed output of 1 and the other neurons have a fixed output of 0. The outputs of the neurons in the last stages are fixed at an intermediate value to represent zero *a priori* knowledge about these states. Notice that the network reaches an equilibrium point in which only the top neurons in each state (representing the "00" node in figure 1) are on and all others are off. This case illustrates that the network can correctly decode an unerrored input and that it does so rapidly, i.e., in about one time constant. In this case, with no errors in the input, the network performs the

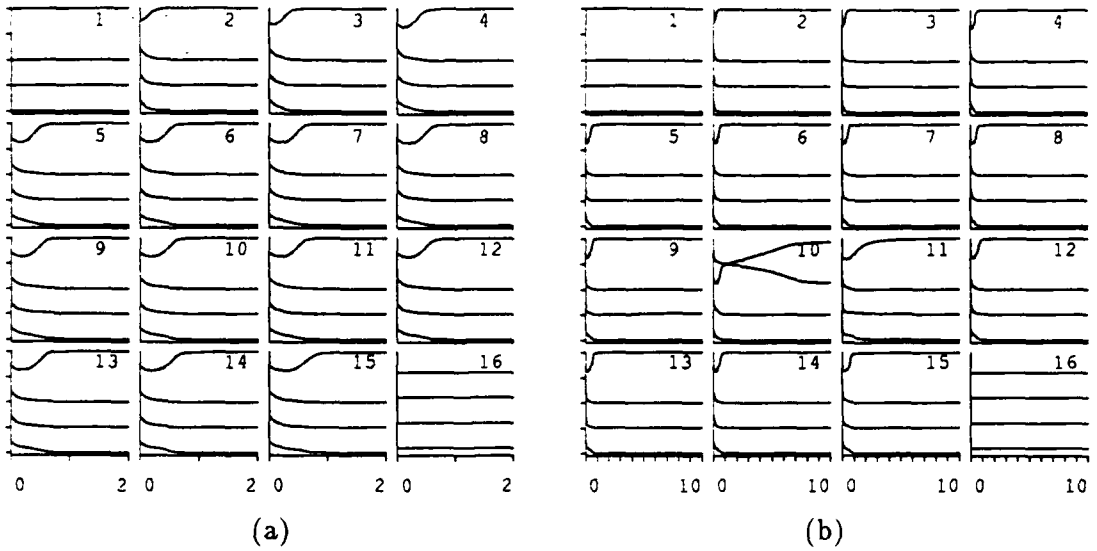

Figure 2: Evolution of the trellis network for (a) unerrored input, (b) input with burst errors: **R** is 000 000 000 000 000 000 000 000 111 000 000 000 000 000 000. $\lambda = 10.$, $A = 1.0$, $B = 1.0$, $C = 0.75$, $T = 0.02$. The initial conditions are $x_{1,1} = 1.$, $x_{1,j} = 0.0$, $x_{16,j} = 0.2$, all other $x_{i,j} = 0.0$.

same function as Hopfield and Tank's network and does so quite well. Although we have not been able to prove it analytically, all our simulations support the conjecture that if $x_{i,j}(0) = \frac{1}{2}$ for all i and j then the network will always converge to the global minimum.

One of the more difficult decoding problems for this network is the correction of a burst of errors in a transition subsequence. Figure 2b shows the evolution of the network when three errors occur in the transition between stages 9 and 10. Note that 10 unit time intervals are shown since complete convergence takes much longer than in the first example. However, the network has correctly decoded many of the stages far from the burst error in a much shorter time.

If the received codeword contains scattered errors, the convolutional decoder should be able to correct more than 3 errors. Such a case is shown in figure 3a in which the received codeword contains 7 errors. The system takes longest to converge around two transitions, 5–6 and 11–12. The first is in the midst of consecutive subsequences which each have one bit errors and the second transition contains two errors.

To illustrate that the energy function shown in equation 13 is a good candidate for a Lyapunov function for this network, it is plotted in figure 3b for the three cases described above. The nonlinearity used in these simulations has a gain of ten, and, as predicted by the large gain limit, the energy decreases monotonically.

To more thoroughly explore the behavior of the network, the simulation program was modified to test many possible error patterns. For one and two errors, the program exhaustively tested each possible error pattern. For three or more errors, the errors were generated randomly. For four or more errors, only those errored sequences for which the MLS estimate was the sequence of all zeros were tested. The results of this simulation are summarized in the column labeled "two-nearest" in figure 4. The performance of the network is optimum if no more than 3 errors are present in the received sequence, however for four or more errors, the network fails to correctly decode some sequences that the MLSE decoder can correctly decode.

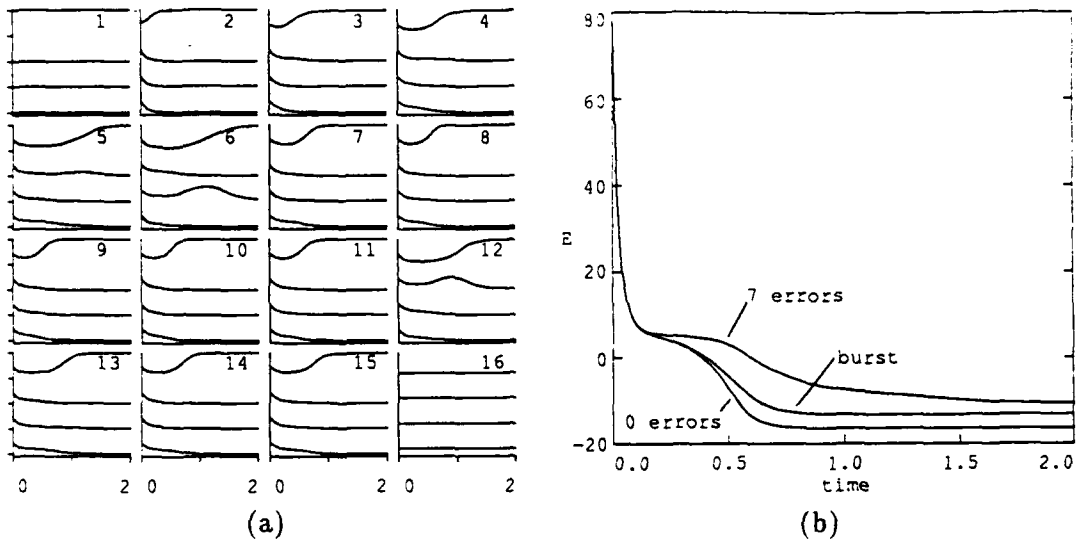

Figure 3: (a) Evolution of the trellis network for input with distributed errors. The input, **R**, is 000 010 010 010 100 001 000 000 000 000 110 000 000 000 000. The constants and initial conditions are the same as in figure 2. (b) The energy function defined in equation 13 evaulated for the three simulations discussed.

errored bits	number of test vectors	number of errors	
		two-nearest	four-nearest
0	1	0	0
1	39	0	0
2	500	0	0
3	500	0	0
4	500	7	0
5	500	33	20
6	500	72	68
7	500	132	103
Total	2500	244	191

Figure 4: Simulation results for a deconvolution network for a $K = 3$, rate 1/3 code. The network parameters were: $\lambda = 15$, $A = 6$, $B = 1$, $C = 0.45$, and $T = 0.025$.

For locally interconnected networks, the major concern is the flow of information through the network. In the simulations presented until now, the neurons in each stage are connected only to neurons in neighboring stages. A modified form of the network was also simulated in which the neurons in each stage are connected to the neurons in the four nearest neighboring stages. To implement this network, the subroutine to initialize the connection weights was modified to assign a non-zero value to $w_{i,j,i+2,k}$. This is straight-forward since, for a code with a constraint length of three, there is a single path connecting two nodes a distance two apart.

The results of this simulation are shown in the column labeled "four-nearest" in figure 4. It is easy to see that the network with the extra connections performs better

than the previous network. Most of the errors made by the nearest neighbor network occur for inputs in which the received subsequences $\mathbf{r}_i$ and $\mathbf{r}_{i+1}$ or $\mathbf{r}_{i+2}$ contain a total of four or more errors. It appears that the network with the additional connections is, in effect, able to communicate around subsequences containing errors that block communications for the two-nearest neighbor network.

7 Summary and Conclusions

We have presented a locally interconnected network which minimizes a function that is analogous to the log likelihood function near the global minimum. The results of simulations demonstrate that the network can successfully decode input sequences containing no noise at least as well as the globally connected Hopfield-Tank [6] decomposition network. Simulations also strongly support the conjecture that in the noiseless case, the network can be guaranteed to converge to the global minimum. In addition, for low error rates, the network can also decode noisy received sequences.

We have been able to apply the Cohen-Grossberg proof of the stability of "on-center off-surround" networks to show that each stage will maximize the desired local "likelihood" for noisy received sequences. We have also shown that, in the large gain limit, the network as a whole is stable and that the equilibrium points correspond to the MLSE decoder output. Simulations have verified this proof of stability even for relatively small gains. Unfortunately, a proof of strict Lyapunov stability is very difficult, and may not be possible, because of the cooperative connections in the network.

This network demonstrates that it is possible to perform interesting functions even if only localized connections are allowed, although there may be some loss of performance. If we view the network as an associative memory, a trellis structured network that contains NS neurons can correctly recall 2^S memories. Simulations of trellis networks strongly suggest that it is possible to guarantee a non-zero minimum radius of attraction for all memories. We are currently investigating the use of trellis structured layers in multilayer networks to explicitly provide the networks with the ability to tolerate errors and replace faulty neurons.

References

[1] M. Cohen and S. Grossberg, "Absolute stability of global pattern formation and parallel memory storage by competitive neural networks," *IEEE Trans. Sys., Man, and Cyber.*, vol. 13, pp. 815–826, Sep.–Oct. 1983.

[2] S. Grossberg, "How does a brain build a cognitive code," in *Studies of Mind and Brain*, pp. 1–52, D. Reidel Pub. Co., 1982.

[3] J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences USA*, vol. 79, pp. 2554–2558, 1982.

[4] J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proceeedings of the National Academy of Science, USA*, vol. 81, pp. 3088–3092, May 1984.

[5] J. McClelland and D. Rumelhart, *Parallel Distributed Processing, Vol. 1.* The MIT Press, 1986.

[6] D. Tank and J. Hopfield, "Simple 'neural' optimization networks: an A/D converter, signal decision circuit and a linear programming circuit," *IEEE Trans. on Circuits and Systems*, vol. 33, pp. 533–541, May 1986.

[7] A. Viterbi and J. Omura, *Principles of Digital Communications and Coding.* McGraw-Hill, 1979.

Proc. 21st Conf. on
Information Sciences and
Systems, Johns Hopkins Univ.
1987, pp. 381-386.

A Locally Interconnected Neural Network for Deconvolution*

Thomas Petsche and Bradley W. Dickinson
Princeton University
Department of Electrical Engineering
Princeton, NJ 08544

Abstract

We have developed a *neural* or *connectionist* network which optimizes a function having the form of the log likelihood function for the output sequence of a binary symmetric channel whose input comes from a convolutional code. The network may be applied to more general FIR deconvolution problems. It requires mainly localized connections and is intended to represent the type of networks which lend themselves to VLSI implementation. Analytical and empirical results on network performance and stability are described.

1 Introduction

One of the often cited problems in trying to implement neural networks, particularly of the Hopfield type [1], in VLSI is that the networks generally require global interconnections. This causes difficulties, requiring long wires to connect elements that are far apart. It would be much easier to design efficient VLSI implementations if connections could be restricted to be between only elements that are "close" together.

While it is possible to impose a locality requirement on a network which might ordinarily have global interconnections, the performance will suffer. For a variant of Hopfield associative memory networks, it has been shown [2] that capacity decreases in proportion to the maximum allowable distance between connected elements. It would be desirable to design networks in such a way that the locality constraint is initially satisfied, preferably exploiting the underlying structure of the problem.

We have developed a network which optimizes a function that has the form of the likelihood function for decoding convolutional codes or more general FIR signal deconvolution. The structure of the network reflects the structure of the trellis representation of the convolutional code and therefore has the desired locality property. The locality of the final network is also enhanced by the choice of neural model used for each element.

*Work supported by the Office of Naval Research through grant N00014-83-K-0577 and by the National Science Foundation through grant ECS84-05460.

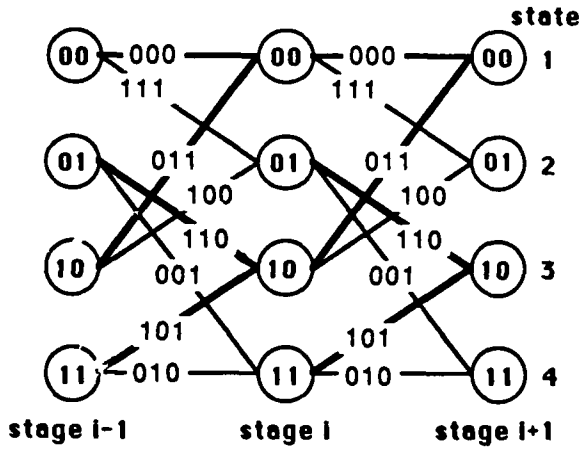

Figure 1: Part of the trellis-code representation for a rate 1/3, $K = 3$ convolutional code.

2 The Network

Consider the trellis graph, shown in figure 1, for a convolutional code that is to be searched by a maximum likelihood estimator such as the Viterbi decoder [3]. For a rate $1/n$ convolutional code with constraint length K, if we force a decision after $5K$ stages, the trellis graph contains $5K$ stages, each containing 2^{K-1} states. The Viterbi algorithm (or any other MLE algorithm) must choose the path through the trellis that has the maximum likelihood of being correct given a (possibly noisy) received bit sequence. Assuming a binary symmetric channel, we can assign a weight to each edge in the trellis graph that is proportional to the number of matching bits in the received bit sequence and the expected sequence for that edge. The maximum likelihood estimate in this case is equivalent to the path with the greatest cumulative weight.

Figure 1 corresponds to a rate 1/3 time invariant convolutional code with a constraint length of 3 where the generator sequences are

$$\mathbf{g}_0 = (1\ 1\ 1) \qquad \mathbf{g}_1 = (1\ 1\ 0) \qquad \mathbf{g}_2 = (0\ 1\ 1).$$

For an input sequence $\mathbf{u} = (u_1, \ldots, u_b, 0, 0)$, the encoder output is $\mathbf{v} = (\mathbf{v}_1, \ldots, \mathbf{v}_{b+2})$. The output after the ith

input bit has entered the encoder is $\mathbf{v}_i = (v_{i1}, v_{i2}, v_{i3})$ where (using modulo-2 addition)

$$\mathbf{v}_i = \sum_{k=\max(1,i-2)}^{i} u_k \mathbf{g}_{i-k}$$

Notice that this code, for a fixed length input containing b bits, converts the b-bit input words into a codeword of length $3(b+2)$ where the 2 is added by introducing two zeros at the end of every input to "zero-out" the code. Equivalently, the coder can be viewed as embedding 2^b memories into a $2^{3(b+2)}$-dimensional space. The minimum distance between valid memories or codewords in this space is the *free distance* of the code, which in this example is 7. This implies that the code is able to correct a *minimum* of three errors in the received signal.

A MLE decoder is designed to accept as input a possibly noisy coded sequence, $\mathbf{r}$, and produce as output the maximum likelihood estimate, $\hat{\mathbf{v}}$, of the original sequence, $\mathbf{v}$. If the set of possible $2(b+2)$-bit encoder output vectors is $\{\mathbf{x}_m : m = 1, ..., 2^{3(b+2)}\}$ and $\mathbf{x}_{m,i}$ is the ith n-bit subsequence of $\mathbf{x}_m$, then

$$\hat{\mathbf{v}} = \arg\max_{\mathbf{x}_m} \prod_{i=1}^{N} p(\mathbf{r}_i | \mathbf{x}_{m,i})$$

In the case of a binary symmetric channel, this is equivalent to

$$\hat{\mathbf{v}} = \arg\max_{\mathbf{x}_m} \sum_{i=1}^{N} M(\mathbf{r}_i, \mathbf{x}_{m,i}) \qquad (1)$$

$M(a, b)$ is the number of matching bits in a and b.

We have defined a neural network which corresponds to the trellis used for the MLE estimation defined above. Each stage of the trellis, representing the set of possible states at each time instant at position i, is implemented as an "on-center off-surround" competitive network [4]. This network will be described in more detail in the next section but for now it suffices to know that it will produce a contrast enhanced version of the input.

The edges in the trellis graph correspond to cooperative connections between neurons. In addition to these cooperative connections, it is sometimes helpful to add inhibitory connections between unconnected nodes in the trellis graph, since these transitions can not occur in the final path. All connections in this model are assumed to be symmetric. The weights assigned to each connection in this model vary with each problem instance; i.e., for each received sequence or subsequence the weights may be different.

More precisely, for a rate $1/n$ code, if there is an edge between nodes $\mathcal{N}_{i,j}$ and $\mathcal{N}_{i+1,k}$ in the graph, and if $(x_{i,j,k,1}, \ldots, x_{i,j,k,n})$ are the encoder output bits for the transition between these two nodes and $(r_{i,1}, \ldots, r_{i,n})$ are the received bits for this transition, then there is a symmetric cooperative connection between these two neurons and the associated weight is

$$m_{i,j,k} = \frac{1}{n} \sum_{l=1}^{n} I_{r_{i,l}}(x_{i,j,k,l}) \qquad (2)$$

If there is no edge between $\mathcal{N}_{i,j}$ and $\mathcal{N}_{i+1,k}$, then $M_{i,j,k} = 0$. $I_a(b)$ is the indicator function:

$$I_a(b) = \begin{cases} 1 \text{ if } a = b \\ 0 \text{ if } a \neq b \end{cases}$$

For more general sequence estimation or FIR signal deconvolution, e.g., MLE sequence estimation in the presence of intersymbol interference, the connection weights would be determined by a similar function. If $\mathbf{x}_{i,j,k}$ and $\mathbf{r}_i$ represent the encoded and received signals, then the connection weight can be any monotonically decreasing function of the distance between the two signals. For a system in which there is no noise corrupting the received signal, analysis of the network is simplified by choosing this function to be a step function which is 1 if $\mathbf{x}_{i,j,k} = \mathbf{r}_i$ and 0 otherwise.

The varying input weights required by this network can be implemented in at least two ways. To obtain a model in the neural network spirit, the received bits would be applied to input neurons whose output is proportional to the degree of match between the expected and actual inputs. The output of these neurons would then modify the signals on the cooperative connections at multiplicative synapses. Such synapses have been observed in biological systems. This method has the advantage of requiring relatively simple neurons for the trellis since their input weights would be fixed. Alternatively, the weights used by each neuron to calculate its output can change with each input. This method is probably the easiest to use for digital implementations and is the one used in our simulations. Observe that in either of these cases, the information required to calculate the weight is local at each edge of the trellis graph and therefore at each connection in the network.

Intuitively, it is easiest to understand the action of the entire network by examining one stage. Consider the nodes in stage i of the trellis graph and assume that the conditional probabilities of the nodes in stages $i-1$ and $i+1$ are known. Then the conditional probability of each node in stage i is simply the sum of the probabilities of each node in $i-1$ and $i+1$ weighted by the transition probabilities. If we look at stage i in the network, and again let the neighboring stages $i-1$ and $i+1$ be fixed with the output of each neuron corresponding to the "likelihood" of the corresponding state at that stage, then the final outputs of the neurons $\mathcal{N}_{i,j}$ will correspond to the "likelihood" of each of the corresponding states. When the stage reaches equilibrium, the neuron corresponding to the most likely state will have the largest output.

3 The Neural Model

In the previous section, we defined the problem to be solved by this network and the connections to be used. These requirements place some restrictions on the neural models that can be used. The model used in this network, called an "on-center off-surround" network because the output of each neuron in the network is used as positive feedback to itself and negative feedback to all the other neurons in the network, was proposed by Grossberg [4]. The model allows the output of each neuron to take on a range of values and was designed to support contrast enhancement and competition. The model also guarantees that the final output of each neuron is a function of the relative intensity of its input as a fraction of the total input provided to the network.

The instantaneous activity, u_i, of each neuron $\mathcal{N}_i$ $(i = 1, \ldots, N)$ in the "on-center off-surround" network is described by a differential equation:

$$\begin{aligned}\dot{u}_i = &-A_i u_i + (B_i - C_i u_i)\,(I_i + f_i(u_i)) \\ &- (D_i u_i + E_i)\Big(\sum_{j=1}^{N} I_j + \sum_{k=1}^{N} F_{i,k}\, g_k(u_k)\Big)\end{aligned} \tag{3}$$

Here A_i, B_i, C_i, D_i, and E_i are constants; $f_i(\cdot)$ and $g_k(\cdot)$ are nonlinear non-decreasing functions; and I_i is an external input to $\mathcal{N}_i$. $F_{i,k}$ is the weight associated with the input to $\mathcal{N}_i$ from $\mathcal{N}_k$. It can be shown that this system restricts u_i in such a way that

$$-\frac{E_i}{D_i} \le u_i \le \frac{B_i}{C_i}.$$

For our deconvolution network, it is not really possible to use equation 3 directly since it is assumes that the external inputs I_i are constant for at least the time it takes the network to converge. To write an equation that is similar to equation 3, however, we define an external input to a neuron in stage i to be any input that does not originate from some neuron in stage i and drop the requirement that the inputs be constant. For simplicity, we also define all the constants to be the same for each neuron and take all the nonlinearities to be equal to the same sigmoid function (spatial homogeneity). Specifically, for the simulations presented in section 5,

$$f_i(x) = g_j(x) = f(x) = \frac{1}{1 + e^{-\lambda x}} \qquad \forall i, j \tag{4}$$

Following Hopfield's notation [5], λ represents the gain of the nonlinearity.

Using the $m_{i,j,k}$ defined in equation 2, the differential equation that governs the instantaneous activity of the neurons in a deconvolution network with S stages and N states in each stage can be written as

$$\begin{aligned}\dot{u}_{i,j} = &-A u_{i,j} \\ &+ (B - u_{i,j})\Big(f(u_{i,j}) + \sum_{k=1}^{N}\big[m_{i-1,k,j} f(u_{i-1,k}) \\ &\qquad + m_{i,j,k} f(u_{i+1,k})\big]\Big) \\ &- (C + u_{i,j}) \sum_{k \ne j}^{N}\Big(f(u_{i,k}) + \sum_{l=1}^{N}\big[m_{i-1,k,l} f(u_{i-1,k}) \\ &\qquad + m_{i,l,k} f(u_{i+1,k})\big]\Big)\end{aligned} \tag{5}$$

Equation 5 can be rewritten more compactly as

$$\dot{u}_{i,j} = -A u_{i,j} - \sum_{k=1}^{S}\sum_{l=1}^{N}\big(u_{i,j} S_{i,j,k,l} f(u_{k,l}) - T_{i,j,k,l} f(u_{k,l})\big) \tag{6}$$

where, for $1 \le l \le N$,

$$\begin{aligned}&S_{i,j,i,l} = 1 \\ &S_{i,j,i-1,l} = \textstyle\sum m_{i-1,l,q} \\ &S_{i,j,i+1,l} = \textstyle\sum_{q}^{q} m_{i,q,l} \\ &T_{i,j,i,j} = B \\ &T_{i,j,i,l} = -C \quad \forall \quad l \ne j \\ &T_{i,j,i-1,l} = B m_{i-1,l,j} - C \textstyle\sum_{q \ne j} m_{i-1,l,q} \\ &T_{i,j,i+1,l} = B m_{i,j,l} - C \textstyle\sum_{q \ne j} m_{i,q,l}\end{aligned} \tag{7}$$

If $k \notin \{i-1, i, i+1\}$, then $S_{i,j,k,l} = T_{i,j,k,l} = 0$.

To eliminate the need for global interconnections within a stage, we can add summing elements to calculate

$$X_i = \sum_{j=1}^{N} f(x_{i,j}) \quad \text{and} \quad J_i = \sum_{j=1}^{N} I_{i,j}$$

where

$$I_{i,j} = \sum_{k=1}^{N}\big[m_{i-1,k,j} f(u_{i-1,k}) + m_{i,j,k} f(u_{i+1,k})\big] \tag{8}$$

Then equation 5 can be rewritten as

$$\dot{u}_{i,j} = -A u_{i,j} + (B + C)\big(f(u_{i,j}) + I_{i,j}\big) - u_{i,j}(X_i + J_i)$$

4 Stability of the Network

At the end of section 2, the desired operation of a single stage given that the neighboring stages are fixed is described. It is possible to show that in this situation a single stage is stable. To do this, fix $f(u_{k,l})$ for $k \in \{i-1, i+1\}$ so that equation 6 can be written in the same form as equation 3:

$$\begin{aligned}\dot{u}_{i,j} = &-A u_{i,j} + (B - u_{i,j})\,(I_{i,j} + f(u_{i,j})) \\ &-(u_{i,j} + C)\Big(\sum_{k=1}^{N} I_{i,k} + \sum_{k=1}^{N} f(u_{i,k})\Big)\end{aligned} \tag{9}$$

where $I_{i,j}$ is defined in equation 8.

Equations 9 and 3 are special cases of the more general nonlinear system

$$\dot{x}_i = a_i(x_i)\left(b_i(x_i) - \sum_{k=1}^{n} c_{i,k} d_k(x_k)\right)$$

where

$a_i(x_i)$ is continuous and $a_i(x_i) > 0$ for $x_i \geq 0$
$b_i(x_i)$ is continuous for $x_i \geq 0$
$c_{i,k} = c_{k,i}$
$d_i(x_i) \geq 0 \quad \forall \quad x_i \in (-\infty, \infty)$

It has been shown [6] that a system that can be written in this form has a global Lyapunov function which can be written

$$V(\mathbf{x}) = -\sum_{i=1}^{n} \int_0^{x_i} b_i(\xi_i) d_i'(\xi_i) d(\xi_i) + \frac{1}{2} \sum_{j=1}^{n} \sum_{k=1}^{n} c_{j,k} d_j(x_j) d_k(x_k) \tag{10}$$

and that, therefore, such a system is asymptotically stable. In our case, this means that a single stage has the desired behavior when the neighboring stages are fixed.

It does not seem possible to use the Cohen-Grossberg stability proof for the entire system in equation 5. Extensions of 3 and 10 also seem to fall short. In fact, Cohen and Grossberg note that networks which allow cooperative interactions define systems for which no stability proof exists [6].

In another approach, Hopfield [5] showed that a network with simpler feedback, governed by

$$\dot{u}_i = \frac{1}{R_i} u_i + \sum_{j=1}^{N} T_{i,j} f_j(u_j)$$

has a Lyapunov function of the form (for $V_i = f_i(x_i)$)

$$E = -\frac{1}{2} \sum_{i,j} T_{i,j} V_i V_j + \sum_i \frac{1}{R_i} \int_0^{V_i} f_i^{-1}(V) dV \tag{11}$$

Hopfield argued that the nonlinearity can be normalized so that we can write

$$\bar{f}^{-1}(V_i) = \lambda x_i$$

where the bar denotes the normalized function. Then the integral in equation 11 can be written

$$\int_0^{V_i} f_i^{-1}(V) dV = \frac{1}{\lambda} \int_0^{V_i} \bar{f}_i^{-1}(V) dV$$

Hopfield used this manipulation to argue that for large gains ($\lambda \to \infty$), the second term in equation 11 is negligible and so the network of analog neurons has the same equilibrium points as a network of discontinuous on-off neurons.

A possible extension of equation 11 for the deconvolution network is

$$E = -\tfrac{1}{2} \sum_{i,j,k,l} T_{i,j,k,l} V_{i,j} V_{k,l} - \sum_{i,j} \frac{1}{\lambda} \left(-A - \sum_{k,l} S_{i,j,k,l} V_{k,l}\right) \int_{\frac{1}{2}}^{V_{k,l}} \bar{f}^{-1}(V) dV \tag{12}$$

The time derivative of E is

$$\dot{E} = -\sum_{i,j} \frac{dV_{i,j}}{dt} \left(-A u_{i,j} - u_{i,j} \sum_{k,l} S_{i,j,k,l} V_{k,l} + \sum_{k,l} T_{i,j,k,l} V_{k,l} - \frac{1}{\lambda} \sum_{k,l} S_{i,j,k,l} \int_{\frac{1}{2}}^{V_{k,l}} \bar{f}^{-1}(V) dV\right) \tag{13}$$

It can be shown that for $f(x) = \left(1 + e^{-\lambda x}\right)^{-1}$,

$$\int_{\frac{1}{2}}^{V_i} \bar{f}_i^{-1}(V) dV = B < \infty$$

In this deconvolution network, $S_{i,j,k,l} = 0$ for $|i-k| > 1$ or $|j-l| > S$, so there are no more than $3S$ terms in the summation. Then, in the limit as $\lambda \to \infty$, the term in parentheses in equation 13 converges to $\dot{u}_i$ in equation 6. Using the chain rule, we can write

$$\lim_{\lambda \to \infty} \dot{E} = -\sum_{i,j} \left(\frac{dV_{i,j}}{dt}\right)^2 \left(\frac{d}{dV_{i,j}} f^{-1}(V_{i,j})\right)$$

It can also be shown that

$$\frac{d}{dV_i} f^{-1}(V_i) \geq 0$$

for all V_i, and this implies that

$$\lim_{\lambda \to \infty} \dot{E} \leq 0 \; \forall \; \mathbf{u}.$$

Therefore, for large gains, E as defined in equation 12 is a Lyapunov function for the system described by equation 5.

We can apply the same asymptotic argument to the energy function in equation 12 since the term on the second line of the equation is also scaled by λ. This implies that the equilibrium points in this network in the large gain limit also correspond to the equilibrium points of a network of discontinuous on-off neurons. For the binary neuron case, it is fairly straight forward to show that the energy function has minima at the desired decoder outputs if we assume that only one neuron in each stage may b on or that B and C are appropriately chosen to favor this. This bound is, however, not as tight as that on the derivative of the energy function since there $O(S^2N)$ terms in the summation rather than $O(S)$ as above.

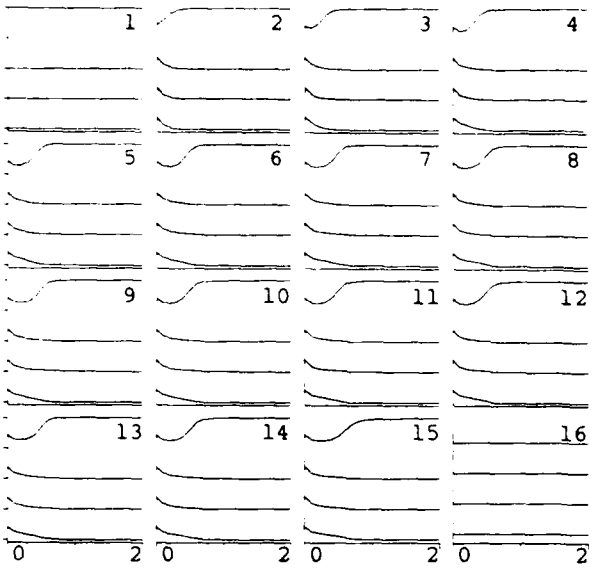

Figure 2: Evolution of the trellis network for unerrored input. $\lambda = 10.$, $A = 1.0$, $B = 1.0$, $C = 0.75$, $T = 0.02$, input is all zeros. The initial conditions are $x_{1,1} = 1.$, $x_{1,j} = 0.0$, $x_{16,j} = 0.2$, all other $x_{i,j} = 0.0$.

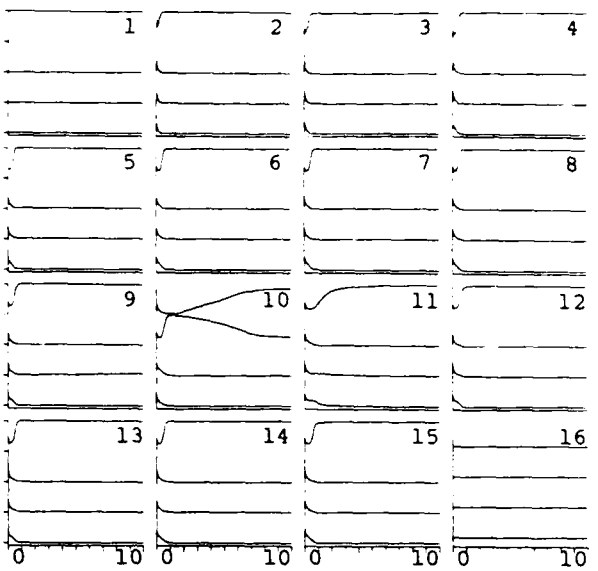

Figure 3: Evolution of the trellis network for input with burst errors. The input is 000 000 000 000 000 000 000 000 111 000 000 000 000 000 000. The constants and initial conditions are the same as in figure 2.

5 Simulation Results

This network was simulated by discretizing equation 5 using Euler's method. For a sampling frequency of $1/T$, the equation of the updated activity, $u_{i,j}(t+1)$, is

$$
\begin{aligned}
u_{i,j}(t+1) &= u_{i,j}(t) - TAu_{i,j}(t) \\
&\quad + T(B - u_{i,j}(t))\Bigg(f(u_{i,j}(t)) \\
&\quad + \sum_{k=1}^{N}\left[m_{i-1,k,j}f(u_{i-1,k}(t)) + m_{i,j,k}f(u_{i+1,k}(t))\right]\Bigg) \\
&\quad - T(C + u_{i,j}(t))\sum_{k\neq j}^{S}\Bigg(f(u_{i,k}(t)) \\
&\quad + \sum_{l=1}^{N}\left[m_{i-1,k,l}f(u_{i-1,k}(t)) + m_{i,l,k}f(u_{i+1,k}(t))\right]\Bigg)
\end{aligned}
$$

The simulations presented here are for the convolutional code illustrated in figure 1. Since this code has a constraint length of 3, there are 4 possible states in each stage and we will use a total of 16 stages. The first and last stages are fixed since we assume that we have prior knowledge or a decision about the first stage and zero knowledge about the last stage. The transmitted codeword is assumed to be all zeros.

Figure 2 shows the evolution of the network over 2 unit time intervals with $T = 0.02$ when the received codeword contains no noise. The output of each stage is a vertical set of 4 curves. The upper-left set is the output of the first stage; the upper-most curve is the output of the first neuron in the stage. For the first stage, the first neuron has a fixed output of 1 and the other neurons have a fixed output of 0. The outputs of the neurons in the last stages are fixed at an intermediate value to represent zero *a priori* knowledge about these states. Notice that the network reaches an equilibrium point in which only the top neurons in each state (representing the "00" node in figure 1) are on and all others are off. This case simply illustrates that the network can correctly decode an unerrored input and that it does so rapidly, i.e., in about one time constant.

One of the more difficult decoding problems for this network is the correction of a burst of errors in a transition subsequence. Figure 3 shows the evolution of the network when three errors occur in the transition between stages 9 and 10. Note that 10 unit time intervals are shown since complete convergence takes much longer than in the first example. However, the network has correctly decoded many of the stages far from the burst error in a much shorter time.

If the received codeword contains scattered errors, the convolutional decoder should be able to correct more than 3 errors. Such a case is shown in figure 4 in which the received codeword contains 7 errors. The system takes longest to converge around two transitions, 5–6 and 11–12. The first is in the midst of consecutive subsequences which each have one bit errors and the second transition contains two errors.

To illustrate that the energy function shown in equation 12 is a good candidate for a Lyapunov function for this network, it is plotted in figure 5 for the three cases

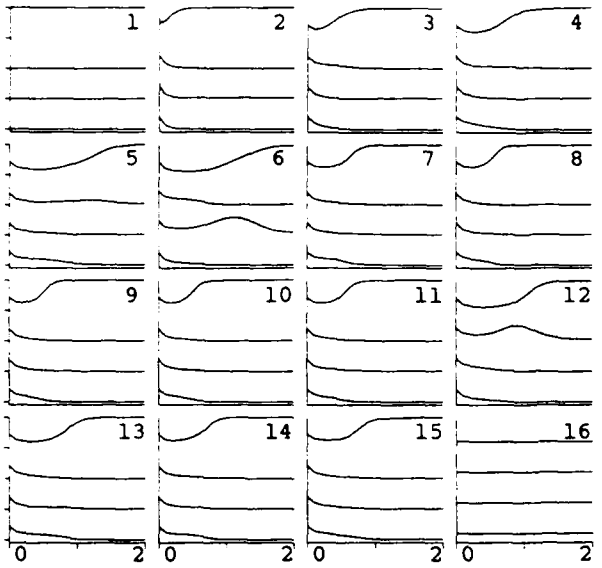

Figure 4: Evolution of the trellis network for input with distributed errors. The input is 000 010 010 010 100 001 000 000 000 000 110 000 000 000 000. The constants and initial conditions are the same as in figure 2.

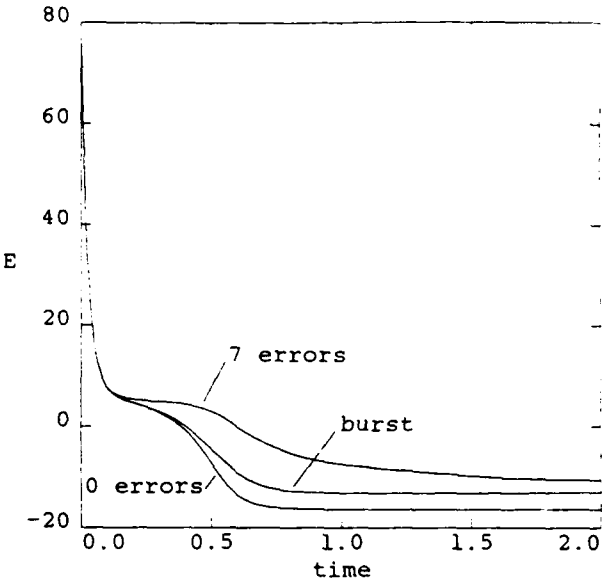

Figure 5: The energy function defined in equation 12 evaluated for the networks whose outputs are shown in figures 2, 3, and 4.

described above. The nonlinearity used in these simulations has a gain of ten, and, as predicted by the large gain limit, the energy decreases monotonically.

6 Discussion

We have presented a network which minimises a function that is analogous to the log likelihood function. The results of several simulations demonstrate that the network can successfully decode noisy input vectors. Other simulations with different codes have supported these results.

Section 4 illustrates that a proof of the stability of "on-center off-surround" networks can be applied to show that each stage will maximise the desired local "likelihood". The same section also showed that, in the large gain limit, the network as a whole is stable and the the equilibrium points correspond to the MLE decoder output.

Our network is distinguished by the fact that connections are localised. Although each stage has been assumed to be globally interconnected, with the addition of simple summing or averaging elements this is not necessary. Each element would receive the sum of the outputs of that stage rather than each individual value. Connections between elements in different stages can occur only if the stages are adjacent. Given the structure of a rate $1/n$ code, no element can be connected to more than 2 elements in each of the two neighboring stages.

References

[1] D.W. Tank and J.J. Hopfield, "Simple 'neural' optimization networks: an a/d converter, signal decision circuit, and a linear programming circuit," *IEEE Trans. Circuits and Systems*, vol. 33, pp. 533–541, May 1986.

[2] A. Kuh, *Stochastic Models for Interacting Systems.* PhD thesis, Princeton University, 1986.

[3] A.J.Viterbi and J.K.Omura, *Principles of Digital Communication and Coding*, p. 227 ff. McGraw-Hill, 1979.

[4] S. Grossberg, "How does a brain build a cognitive code," in *Studies of Mind and Brain*, pp. 1–52, D. Reidel Pub. Co., 1982.

[5] J.J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proc. Nat. Acad. Sci., USA*, vol. 81, pp. 3088–3092, May 1984.

[6] M.A. Cohen and S. Grossberg, "Absolute stability of global pattern formation and parallel memory storage by competitive neural networks," *IEEE Trans. Sys., Man, and Cyber.*, vol. 13, pp. 815–826, Sep.–Oct. 1983.

Information Capacity of Associative Memories

ANTHONY KUH, MEMBER, IEEE, AND BRADLEY W. DICKINSON, FELLOW, IEEE

Abstract — Associative memory networks, consisting of highly interconnected binary-valued cells, have been used to model neural networks. Tight asymptotic bounds have been found for the information capacity of these networks. We derive the asymptotic information capacity of these networks using results from normal approximation theory and theorems about exchangeable random variables.

I. Introduction

FOR MANY YEARS researchers in various disciplines have studied models for the brain. Many models have been developed in attempts to understand how neural networks function. One class of such models is based on the concept of associative memory [1]–[27]. Associative memories are composed of a collection of interconnected elements having data storage capabilities. The elements are accessed in parallel by a data probe vector rather than by a set of specific addresses [14].

Recent years have seen interest increasing in the modeling of neural networks for possible applications to computer architectures. Associative memory network (AMN) models of one particular form, consisting of highly interconnected threshold devices [1], [2], [5]–[12], [24]–[27] have received much attention. These models are sometimes referred to as binary associative memory networks (BAMN's).

This paper discusses some analytical aspects of the BAMN models. Specifically, we analyze the storage capabilities of these models. We consider the case where cells can take on only one of two values $\{-1,1\}$. Our work is motivated by a desire to understand better the results of [12], where various elaborate arguments are used to find the asymptotic value of the network storage capacity. In contrast, we determine the asymptotic network storage capacity by applying normal approximation theory and theorems about exchangeable random variables. This new approach contributes to a better understanding of the results and provides a means of extending the analysis to more general AMN models.

Manuscript received January 26, 1988; revised April 13, 1988. This work was supported in part by the Office of Naval Research under Grant N00014-83-K-0577, in part by the National Science Foundation under Grants ECS84-05460 and MIP-8710868, and in part by the Pacific International Center for High Technology Research. This work was partially presented at the 1986 IEEE International Symposium on Information Theory, Ann Arbor, MI.

A. Kuh is with the Department of Electrical Engineering, University of Hawaii at Manoa, Honolulu, HI 96822.

B. W. Dickinson is with the Department of Electrical Engineering, Princeton University, Princeton, NJ 08544.

IEEE Log Number 8825711.

In Section II the standard BAMN model and the notion of capacity of the network are described. The operation and the construction of the standard model are discussed. In this model each updating operation on a cell is performed by thresholding a linearly weighted sum of other cell values. If the threshold is exceeded the cell takes on a 1 value; otherwise, it takes on a -1 value. A network is characterized by a matrix of weights that determines the strengths of the interconnections between different cells. The weight matrix is constructed from a sum of outer products of vectors chosen to be the desired "codewords" to be stored by the network.

The information capacity of the standard model is derived in Section III. This requires a formalization of the notions of capacity and stability. Following [12] we consider two different definitions. In the first, capacity is related to the maximum number of codewords that can be used to construct the AMN while maintaining a fixed codeword as a stable vector. In the second definition, the capacity is related to the largest number of codewords that can all be stored as stable vectors in the network. We also consider the radius of attraction of each of these codewords. For example, if the state of the AMN after a few update operations converges to a given vector for all initial probe vectors at a Hamming distance of K or less, then the given stable vector has a radius of attraction of at least K. Proofs of the various results are given in the Appendices. They involve normal approximation theory and theorems from exchangeable random variables. Finally, in Section IV we summarize the main theoretical results of this paper and introduce extensions for further research.

II. Operation and Construction of the Binary Associative Memory Network Model

In this section we discuss the AMN model presented in [1], [2], sometimes referred to as the binary associative memory network. A network consists of cells $\{X_i\}$, $1 \le i \le N$, with each cell taking on one of the values $\{-1,1\}$. Each cell affects all other cells through an interconnection or weight matrix T. The interconnection matrix is symmetric with 0 values on its diagonal. Each cell is updated at random with the update events forming a Poisson process with rate λ. At each update the linearly weighted sum of all other cell values is compared to a given threshold. If the weighted sum exceeds the threshold, the cell takes on a 1 value; if not, the cell takes on a -1 value. We assume that the updating processes of all cells are independent, so that the total number of updates is a Poisson process with rate

0018-9448/89/0100-0059$01.00

$N\lambda$. Using a counter k that is incremented every time any cell is updated, an update of cell i at time $k+1$ is described by the equation

$$X_i(k+1) = \mu\left[\sum_{j \neq i} X_j(k)T(i,j), X_i(k)\right] \quad (2.1)$$

where

$$\mu(x,y) = \begin{cases} 1, & x > 0 \\ y, & x = 0 \\ -1, & x < 0 \end{cases}.$$

Here we have chosen a 0 threshold.

For an AMN with interconnection matrix T, we define a binary N vector V to be invariant if, when V is input into the network, all updates leave the state of the network unchanged. An invariant vector is also called a *stable vector* of the AMN. The set of all stable vectors is denoted by M.

Now consider the construction of an AMN, a process that can be viewed as learning. The T matrix is constructed so that certain vectors are stored in the network. A vector is successfully stored in the network if it can be retrieved by an appropriate data probe vector. We let $V(i)$, $1 \le i \le m$ be the codewords, binary N vectors, used to construct the T matrix. The desired behavior of the model when some vector $\hat{V}$ is input into the network, i.e., when the network is initialized at $\hat{V}$, is that after a few updates, the state of the network should become V, a stable vector which is close to $\hat{V}$ in Hamming distance. Several techniques can be used to construct the T matrix of the AMN. Here we use a simple technique involving correlation, which contrasts with techniques using eigenvectors and orthogonal learning approaches shown in [14], [18], [24]; the latter are more complicated to implement. The correlation technique constructs the T matrix from $\{V(i)\}$ as follows. Let

$$T_i = V(i)V(i)^T - I, \qquad 1 \le i \le m \quad (2.2)$$

and then take

$$T = \sum_{i=1}^{m} T_i. \quad (2.3)$$

Hopefully, all of the chosen codewords $\{V(i)\}$ will be stable vectors of the network; however, this cannot occur when m becomes too large in comparison to N. The set of all codewords that are stable vectors is called M_c. Thus $M_c \subset M$, but M also contains the "one's complement" of vectors in M_c and possibly other vectors which we call spurious stable vectors.

To find the capacity of these networks, random coding arguments are used; each component of each codeword is assumed to be chosen independently of all other components, with the probability of a 1 or -1 each equal to $1/2$. Then, given m randomly chosen codewords, one can find the probability that any codeword or that all the chosen codewords are members of M. Two definitions for capacity which we call $\overline{m}(\epsilon)$ and $\hat{m}(\epsilon)$ are introduced in [12]. Before presenting these definitions, we define a synchronous update $\Delta(V)$ as a simultaneous update on all N cells with V initially input into the network. It is easily shown that if V is a stable vector then $V = \Delta(V)$; the converse is not necessarily true. This stronger condition is used in defining $\overline{m}(\epsilon)$ as

$$\overline{m}(\epsilon) = \max m \ni \Pr(V(k) = \Delta(V(k))) > 1 - \epsilon \quad (2.4)$$

(where we may take $k = 1$) and $\hat{m}(\epsilon)$ as

$$\hat{m}(\epsilon) = \max m \ni \Pr(V(i) = \Delta(V(i)), 1 \le i \le m) > 1 - \epsilon. \quad (2.5)$$

We show in Section III that $\overline{m}(\epsilon) \approx N/2\log N$ and that $\hat{m}(\epsilon) \approx N/4\log N$ for any $\epsilon > 0$ and for N sufficiently large; these results were obtained by different, more tedious methods in [12].

We first present a simple example to help visualize how these AMN models work. Take $N = 4$ and $m = 3$, and let

$$V(1) = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \qquad V(2) = \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix} \qquad V(3) = \begin{bmatrix} 1 \\ 1 \\ 1 \\ -1 \end{bmatrix}.$$

Using the correlation method, T is easily found to be

$$T = \begin{bmatrix} 0 & 1 & 3 & -1 \\ 1 & 0 & 1 & 1 \\ 3 & 1 & 0 & -1 \\ -1 & 1 & -1 & 0 \end{bmatrix}.$$

We note that only $V(3) \in M$, (i.e., $M_c = \{V(3)\}$). Fig. 1 shows a diagram of this model.

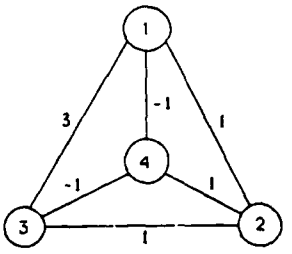

Fig. 1. Example network with edges representing interconnection weights and nodes representing cells.

Before concluding this section we note that in our analysis we always assume that any initial state will converge to a stable vector. This was justified by Hopfield in [2] by noting that

$$E = -\frac{1}{2}\sum_i \sum_j T(i,j)X_i(k)X_j(k), \qquad k \ge 0 \quad (2.6)$$

is a monotonic decreasing function of the update counter k and that the elements of M correspond to the local minima of E. Therefore, any initial state will converge to a stable vector.

III. Derivation of the Information Capacity for the BAMN Model

This section studies the capacities $\overline{m}(\epsilon)$ and $\hat{m}(\epsilon)$ using results from normal approximation theory and theorems about exchangeable random variables. Our main results

are asymptotic expressions for these quantities. We discuss the two patterns for convergence of initial states to some stable vector. We also consider the error-correcting capabilities or the radius of attraction of each codeword. Much of the original discussion about capacity, convergence, and radius of attraction can be found in [12] and [24]. The key differences in our approach versus that of [12] are the proofs of the main theorems; these are given in the appendices.

Associated with each cell value we define the interaction strength (IS) of cell j for codeword k as

$$U(j,k) = \sum_{i \neq j} T(j,i)V_i(k) = \sum_{i \neq j} \sum_{l=1}^{m} V_i(l)V_j(l)V_i(k)$$
$$= (N-1)V_j(k) + \sum_{i \neq j} \sum_{l \neq k} V_i(l)V_j(l)V_i(k). \tag{3.1}$$

According to the standard model, when a cell is updated its next value is determined from a comparison of its IS with a threshold value. Using the random coding model described in the previous section $U(j,k)$ is a random variable. We transform this random variable by letting

$$u(j,k) = \sum_{i \neq j} \sum_{l \neq k} V_i(l)V_j(l)V_i(k)V_j(k) = \sum_{l \neq k} u_j(l)$$
$$= \left[U(j,k) - (N-1)V_j(k)\right]V_j(k) \tag{3.2}$$

where we call $u(j,k)$ the normalized interaction strength (NIS). The $u_j(l)$ for $l \neq k$ are random variables having probability mass function identical to a shifted binomial random variable with mean 0, $N-1$ points, and parameter $p = 1/2$. Since $V(l)$ are chosen independently for all l, the $u(j,k)$ are random variables having probability mass function identical to a shifted binomial random variable with mean 0, $(N-1)(m-1)$ points, and parameter $p = 1/2$.

To evaluate $\bar{m}(\epsilon)$ we need to compute

$$\bar{p} = \Pr(V(k) = \Delta(V(k)))$$
$$= \Pr\left(\bigcap_{j=1}^{N} \{u(j,k) \geq -N+1\}\right) \tag{3.3}$$

(where we may take $k = 1$). To evaluate $\hat{m}(\epsilon)$ we need to compute

$$\hat{p} = \Pr(V(k) = \Delta(V(k)), 1 \leq k \leq m)$$
$$= \Pr\left(\bigcap_{j=1}^{N} \bigcap_{k=1}^{m} \{u(j,k) \geq -N+1\}\right) \tag{3.4}$$

The major stumbling block to analyzing (3.3) and (3.4) is the fact that the $u(j,k)$ are not independent. In fact, it is easily shown that

$$E\{u(j,k)u(l,m)\} = \begin{cases} (N-1)(m-1), & j = l,\ k = m \\ m-1, & j \neq l,\ k = m \\ N-1, & j = l,\ k \neq m \\ 1, & \text{otherwise.} \end{cases} \tag{3.5}$$

By the Demoivre–Laplace theorem [28], for large N, $u(j,k)$ converges in distribution to a Gaussian random variable with the same first- and second-order moments. Thus we let $g(j,k)$ be a Gaussian random variable with the same first- and second-order moments as $u(j,k)$ and investigate the quantities

$$\bar{p}_G = \Pr\left(\bigcap_{j=1}^{N} \{g(j,k) \geq -N+1\}\right) \tag{3.6}$$

and

$$\hat{p}_G = \Pr\left(\bigcap_{j=1}^{N} \bigcap_{k=1}^{m} \{g(j,k) \geq -N+1\}\right). \tag{3.7}$$

Let $Q(x) = (1/\sqrt{2\pi})\int_x^\infty e^{x^2/2}\,dx$ be the standard normal error function and $I(\cdot)$ be the standard indicator function. In Appendix I we develop and use the theory of exchangeable random variables to show the following result.

Theorem 1: 1) If

$$m < \frac{N}{\log N} \qquad X_N = \sum_{j=1}^{N} I(g(j,k) \leq -N)$$

and Y_N is a Poisson random variable with parameter $\lambda_1 = NQ(\sqrt{N/m})$, then $X_N \to Y_N$ in distribution.

2) If

$$m \leq \frac{N}{2\log N} \qquad X_N = \sum_{j=1}^{N} \sum_{k=1}^{m} I(g(j,k) \leq -N)$$

and Y_N is a Poisson random variable with parameter $\lambda_2 = NmQ(\sqrt{N/m})$, then $X_N \to Y_N$ in distribution.

We can use this theorem to evaluate the capacities defined earlier. Letting $m = N/a\log N$ we solve for the constant a for both cases. From the theorem, for large N we have

$$\bar{p}_G = e^{-NQ(\sqrt{N/m})} \tag{3.8}$$

and

$$\hat{p}_G = e^{-NmQ(\sqrt{N/m})}. \tag{3.9}$$

By repeated integration by parts we obtain the expansion

$$Q(x) = \frac{e^{-x^2/2}}{x\sqrt{2\pi}}\left[1 + \sum_{i=1}^{\infty} \frac{(-1)^i}{x^{2i}} \prod_{j=1}^{i} (2j-1)\right].$$

Hence, for large x, we have the approximation $Q(x) \approx e^{-x^2/2}/x\sqrt{2\pi}$. Then we have the following result.

Theorem 2: For case 1) in Theorem 1, assuming $\bar{p} \approx \bar{p}_G$, $\bar{p}_G \gg e^{-N}$, and as $N \to \infty$,

$$\bar{m}(1-\bar{p}_G) \approx \frac{N}{2\log N - 2\log\left(\log\left(\frac{1}{\bar{p}_G}\right)\right) - \log(\log N) - \log 4\pi}$$
$$\approx \frac{N}{2\log N}. \tag{3.10}$$

For case 2) in Theorem 1, assuming $\hat{p} \approx \hat{p}_G$, $\hat{p}_G \gg e^{-N/2}$, and as $N \to \infty$,

$$\hat{m}(1-\hat{p}_G) \approx \frac{N}{4\log N - 2\log\left(\log\left(\frac{1}{\hat{p}_G}\right)\right) - 3\log(\log N) - \log 128\pi} \approx \frac{N}{4\log N}. \tag{3.11}$$

Next we consider conditions under which $\bar{p}_G \to \bar{p}$ and $\hat{p}_G \to \hat{p}$. Appendix II shows that when N is sufficiently large and $m = N/a\log N$ for $1/2 < a \ll N/\log N$ then $\bar{p}_G \to \bar{p}$. This is a consequence of normal approximation theory. It can similarly be shown that $\hat{p}_G \to \hat{p}$. Therefore, for any $\epsilon > 0$ we can find an N large enough such that $\bar{m}(\epsilon) \approx N/2\log N$ and $\hat{m}(\epsilon) \approx N/4\log N$.

In Fig. 2 we plot some simulation results for $\bar{p}$ as a function of m/N; we also plot the theoretical graph of $\bar{p}_G$ versus m/N. For $N = 16$ the simulations and the Gaussian approximations differ substantially, but for $N = 256$ the simulations and the Gaussian approximations are almost identical. The simulations were performed by choosing random codewords, updating the T matrix, and then checking if $V(1) \in M$. This process is continued until the number of codewords, $m = O(N)$. This gives a simulation of one sample network and is shown in Fig. 3. From Monte Carlo simulations of several sample networks, the value of $\bar{p}$ for a given m is easily found.

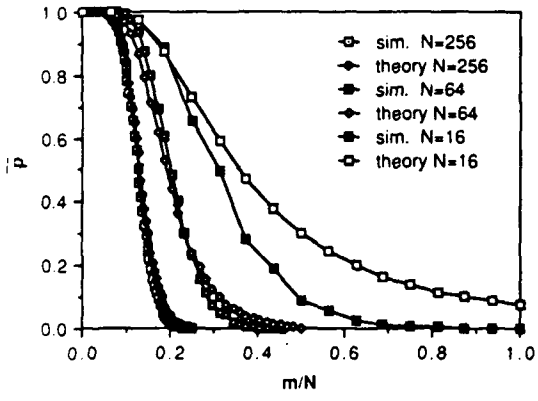

Fig. 2. Simulations comparing theoretical Gaussian approximation $\bar{p}_G$ to Monte Carlo simulations of $\bar{p}$ for $N = 16$, $N = 64$, and $N = 256$.

To this point our assumption has been that we input a codeword with no errors and then calculate the probability that the state of the system does not change after any update. In AMN we are also interested in recovering stored patterns even when some information about the data is lost. For a fixed AMN, we say that a codeword V has K error correcting ability if all vectors $\hat{V}$ within Hamming distance K are correctable by one synchronous update, that is, $V = \Delta(\hat{V})$. We can then evaluate the capacity of the network if we require all codewords to be K error correcting with high probability or if we require a typical codeword to be K error correcting. We define these two capacities as $\hat{m}(\epsilon, K)$ and $\bar{m}(\epsilon, K)$, respectively.

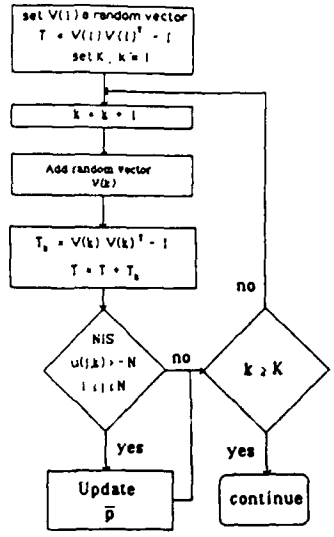

Fig. 3. Algorithm used for conducting simulations of $\bar{p}$.

Using the same type of arguments as in the first part of this section, we first find the NIS:

$$u(j,k,\hat{V}) = \sum_{i \neq j} \sum_{l \neq k} V_i(l)V_j(l)\hat{V}_i(k)\hat{V}_j(k) = \sum_{l \neq k} u_j(l). \tag{3.12}$$

From (3.2), we note that $u(j,k,\hat{V})$ has the same distribution as $u(j,k)$. Observe that

$$N - 1 - 2K = \sum_{i \neq j} V_i(k)V_j(k)\hat{V}_i(k)\hat{V}_j(k) \tag{3.13}$$

when $h(V(k), \hat{V}(k)) = K$ (where $h(x, y)$ is the Hamming distance between x and y). We then define the quantities

$$\bar{p}(K) = \Pr\left(V(k) = \Delta(\hat{V}) \middle| h(V(k), \hat{V}) \le K\right) = \Pr\left(\bigcap_{j=1}^{N} \{u(j,k,\hat{V}) \ge -N+1+2K\}\right), \quad 1 \le k \le m \tag{3.14}$$

and

$$\hat{p}(K) = \Pr\left(V(k) = \Delta(\hat{V}(k)) | h(V(k), \hat{V}(k)) \le K, 1 \le k \le m\right) = \Pr\left(\bigcap_{j=1}^{N} \bigcap_{k=1}^{m} \{u(j,k,\hat{V}) \ge -N+1+2K\}\right) \tag{3.15}$$

in analogy to (3.3) and (3.4). By working with the corresponding Gaussian quantities $\bar{p}_G(K)$ and $\hat{p}_G(K)$, Theorem A2 and normal approximation theory give the following result.

Theorem 3: Asymptotically, as $N \to \infty$ and for all $K < N/2$

$$\bar{m}_K(\epsilon) \approx \frac{(N-2K)^2}{2N\log N} \tag{3.16}$$

and

$$\hat{m}_K(\epsilon) \approx \frac{(N-2K)^2}{4N \log N}. \tag{3.17}$$

An AMN can therefore be expected to have storage capabilities and an error correcting ability for any $K < N/2$.

Before concluding this section, we discuss some very simple arguments that can be used to show that the $\bar{m}(\epsilon)$ is at least $N/2\log N$. By using subadditivity of probability measures, we can find a lower bound on $\bar{m}(\epsilon)$. Note that

$$\begin{aligned} 1-\bar{p} &= \Pr\left(\bigcup_{j=1}^{N} (u(j,m) < -N+1)\right) \\ &\le \sum_{j=1}^{N} \Pr(u(j,k) < -N+1) \\ &= N \Pr(u(1,k) < -N+1). \end{aligned} \tag{3.18}$$

Using the De Moivre-Laplace theorem [28], for large N, $\Pr(u(1,k) < -N+1) \approx Q(\sqrt{N/m})$. Therefore, as $N \to \infty$, if the number of codewords m is no larger than $N/2\log N$, then $\bar{p} \to 1$. This lower bound on the probability is relatively tight for $m \le N/2\log N$ but quite poor for values of m that are larger. Using Bonferroni's inequalities [28], we can also upper-bound $\bar{m}(\epsilon)$ by lower-bounding the error term as

$$\begin{aligned} 1-\bar{p} \ge N \Pr(u(1,k) < -N+1) - \frac{N(N-1)}{2} \\ \cdot \Pr(u(1,k) < -N+1, u(2,k) < -N+1). \end{aligned} \tag{3.19}$$

Again, this bound is relatively tight for $m \le N/2\log N$ but blows up for values of m that are larger. Fig. 4 shows a plot of Monte Carlo simulations of $1-\bar{p}$ versus m when $N = 64$. Approximate upper and lower bounds corresponding to the inequalities in (3.18) and (3.19) and computed from normal approximations are also shown.

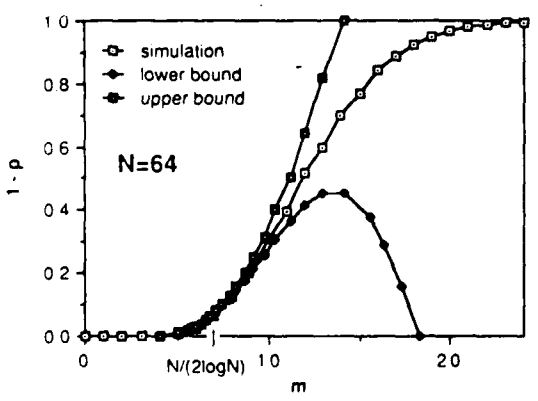

Fig. 4. Simple bounds using Gaussian approximations for $1-\bar{p}$. Lower bound involves calculation of univariate Gaussian distributions and upper bound involves calculation of bivariate Gaussian distributions. Case considered is $N = 64$ and Monte Carlo simulations are used.

IV. Summary and Discussion

This paper has studied the information storage capacity of associative memory network models, in particular the BAMN. Using normal approximation theory and theorems from exchangeable random variables, we proved some asymptotic results about the capacity of the network. These theoretical results were compared to simulations; for $N > 64$ the theoretical and simulation results compare quite favorably.

In proving the asymptotic capacity of the BAMN, we introduced the Gaussian random variables $\{g(j,k),\ 1 \le j \le m,\ 1 \le k \le N\}$ which have the same first- and second-order moments as $\{u(j,k),\ 1 \le j \le m,\ 1 \le k \le N\}$. We then showed that both the set of events $\{\{g(j,k) < x_N\},\ 1 \le k \le N\}$ and $\{\{g(j,k) < x_N\},\ 1 \le j \le m,\ 1 \le k \le N\}$ satisfied Theorem A2, where x_N is some prespecified set of numbers. By using normal approximation theory, we subsequently showed that $\bar{p}_G \to \bar{p}$. It can be similarly shown that $\hat{p}_G \to \hat{p}$. Knowing $\bar{p}$ and $\hat{p}$ the capacity is easily found by using definitions in Section II and III. A more direct proof would be to show that both the set of events $\{\{u(j,k) < x_N\},\ 1 \le k \le N\}$ and $\{\{u(j,k) < x_N\},\ 1 \le j \le m,\ 1 \le k \le N\}$ satisfy Theorem A2. We conjecture that this is indeed true, and proving it is a topic for further research.

Another direction for further research is to consider more ubiquitous AMN models. It would be desirable for these models to retain much of the simple structure of the BAMN, while incorporating such features as random update operations, spatially varying interactions, and more complex learning algorithms. In [27], studies of AMN architectures with these features are described. Using the techniques developed in this paper, we have found asymptotic bounds for the capacity of some random update and spatially varying models. Simulations have confirmed the validity of analytical work. One key result that we have shown is the following: for a class of homogeneous spatially varying models, the capacity of the network is directly proportional to the interconnectivity of the network. Present research is focusing on the determination of the capacity of various AMN, subject to constraints on network interconnectivity.

Appendix I

This Appendix discusses some basic results concerning exchangeable random variables and proves Theorem 1 of Section III. Much of the discussion comes directly from [30]–[33]. Theorem A1 is from [30] and Theorem A2 is from [32].

The problem of interest involves finding the distribution of the minimum of a number of random variables; we therefore begin our discussion with this problem. Let $\{X_i,\ 1 \le i \le N\}$ be identically distributed random variables with the same marginal distribution as X. To find the distribution of $Z = \min_{1 \le i \le N} (X_i)$, we note that

$$\Pr(Z \le z) = 1 - \Pr(X_i > z, 1 \le i \le N).$$

If the $\{X_i\}$ are independent, then

$$\Pr(Z \le z) = 1 - (\Pr(X > z))^N.$$

Let us consider a weaker condition on the $\{X_i\}$. We say that random variables $\{X_i, 1 \le i \le N\}$ are *exchangeable* if their joint distribution is invariant under permutations of the random variables. We also define events $\{C_i, 1 \le i \le N\}$ as exchangeable if, for all choices of indices $1 \le i_1 < \cdots < i_k \le N$, we have

$$\Pr(C_{i_1}, C_{i_2}, \cdots, C_{i_k}) = \alpha_k, \qquad 1 \le k \le N. \tag{A.1}$$

Note that α_k depends only on k and not on the indices i_j. α_k will be referred to as the kth De Finetti constant with $\alpha_0 = 1$. Denoting the event $\{X_i > x\}$ by C_i, if $\{X_i, 1 \le i \le N\}$ are exchangeable random variables, then $\{C_i, 1 \le i \le N\}$ are exchangeable events. De Finetti [34] proved an important theorem relating exchangeable random variables to conditional distributions.

Theorem A1: Let $A(N)$ be the number of $C_i = \{X_i > x\}$ that occur for exchangeable random variables $X_i, 1 \le i \le N$. Then

$$\xi = \lim_{N \to \infty} \frac{A(N)}{N}$$

exists almost surely, and if π (a probability measure on the interval $[0,1]$) is the distribution function of ξ, then

$$\alpha_k = \int_0^1 x^k \pi(dx), \qquad k \ge 0.$$

This result was later extended by several people [35]–[36], giving the following corollary.

Corollary: Given the above conditions there exists a random variable λ such that

$$\Pr(C_{i_1}, \cdots, C_{i_k} | \lambda) = \xi^k$$

where $\xi = \lambda$ *almost surely.*

The above results deal with the limiting case as $N \to \infty$, but we are concerned primarily with finite N. Kendall [30] generalized the above theorems for N finite. As a simple analogy, $N = \infty$ can be viewed as picking marbles from a collection of marbles in an urn and replacing the picked marbles, whereas N finite can be viewed as performing the same operation without replacement. If we let $\delta(\alpha_k) = \alpha_k - \alpha_{k+1}$, $1 \le k \le N$ and $\delta^j(\alpha_k) = \delta(\delta^{j-1}(\alpha_k))$ then

$$\alpha_m = \sum_{s=0}^{N-m} \binom{N-m}{s} \delta^s(\alpha_{N-s}), \qquad 0 \le m \le N. \tag{A.2}$$

If we let $\omega_j = \binom{N}{j} \delta^{N-j}(\alpha_j)$ for $0 \le j \le N$, then ω is a probability distribution since

$$\delta^r \alpha_{N-r} \ge 0, \qquad 0 \le r \le N \tag{A.3}$$

and

$$\sum_{r=0}^{N} \omega_r = 1. \tag{A.4}$$

Then we have

$$\alpha_m = \sum_{s=m}^{N} \frac{\binom{s}{m}}{\binom{N}{m}} \omega_s \tag{A.5}$$

for $0 \le m \le N$ and

$$\Pr(C_{i_1}, C_{i_2}, \cdots, C_{i_k} | W) = \frac{\binom{W}{k}}{\binom{N}{k}} \tag{A.6}$$

where W is a random variable with

$$\Pr(W = j) = \omega_j. \tag{A.7}$$

If we can find the distribution of W, we can find the order statistics of $\{X_i, 1 \le i \le N\}$ for N finite. This distribution is not easy to determine, but Galambos [32] and Kendall [30] have obtained some results for the limiting case under some mild restrictions.

Theorem A2: Let $\{X_i, 1 \le i \le n\}$ be random variables with corresponding distribution functions $\{F_i(x), 1 \le i \le n\}$. Let $\{x_n\}$ be a sequence of real numbers such that

$$\lim_{n \to \infty} \sum_{i=1}^{n} F_i(x_n) = b$$

with $0 < b < \infty$. Setting

$$\alpha_k(n) = \binom{n}{k}^{-1} \sum \Pr(X_{i_1} \le x_n, \cdots, X_{i_k} \le x_n)$$

where the sum is over all indices $1 \le i_1 < i_2 < \cdots < i_k \le n$. Assume that for $n > n_0$ there exists $\alpha_j(n)$, $n < j < M_n$ such that sequence $\alpha_j(n)$, $1 \le j < M_n$ can be associated with a set of M_n exchangeable events. If $M_n = \infty$ for $n > n_0$ or if both $M_n/n \to \infty$ and $n \to \infty$ with

$$\lim_{n \to \infty} n^2 \alpha_2(n) = b^2,$$

then for any j,

$$\lim_{n \to \infty} \Pr(X_j^* < x_n) = \sum_{k=0}^{j} \frac{b^k e^{-b}}{k!}$$

where X_j^* is the jth-order statistic.

The above theorem has the following intuitive interpretation. We construct M_n exchangeable events from some set of n random variables with $M_n \gg n$. If the events are nearly pairwise independent, then l of these events are nearly jointly independent for $2 \le l \le n$. A proof of this theorem presented in [30] is based on finding the characteristic function of $K_n = \sum_{i=1}^{n} I(X_i < x_n)$ and approximating this with the characteristic function of a Poisson random variable with parameter b. The approximation uses Chebyshev's inequality and depends only on the quantities $|n\alpha_1(n) - b|$ and $|n^2\alpha_2(n) - b^2|$ and not on b.

In the problem of interest in Section III, for some N and m we want to find $\Pr(\min(g(i,j), 1 \le i \le N) < -N+1)$ and $\Pr(\min(g(i,j), 1 \le i \le N, 1 \le j \le m) < -N+1)$ where the $\{g(i,j)\}$ are Gaussian exchangeable random variables. We first normalize these random variables, obtaining random variables $\{G_n(i,j)\}$ with the following second moments:

$$E[G_n(i,j)G_n(k,l)] = \begin{cases} 1, & i = k,\ j = l \\ \dfrac{1}{n-1}, & i \ne k,\ j = l \\ \dfrac{1}{m-1}, & i = k,\ j \ne l \\ \dfrac{1}{(n-1)(m-1)}, & \text{otherwise.} \end{cases}$$

We want to find the values of m where Theorem A2 can be applied to evaluate the two problems of Section III. To use

Theorem A2 to evaluate

$$\Pr(\min(g(i,j), 1 \le i \le N) < -N+1) = \Pr(\min(G_N(i,j), 1 \le i \le N) < x_N),$$

the following two conditions must hold for all $n \ge N$:

1) $\lim_{n\to\infty} n^2\alpha_2(n) = b^2$;
2) there exists M_n such that $M_n/n \to \infty$ as $n \to \infty$ with the set $\{G_n(i,j), 1 \le i \le n\}$ augmented to $1 \le i \le M_n$ elements so that all $G_n(i,j)$ are still exchangeable.

Here b is determined by fixing N and setting $b = NQ(-x_N)$ where $x_N = -\sqrt{(N-1)/(m-1)}$. x_n are chosen such that $n\alpha_1(n) = b$ and therefore $x_n = -Q^{-1}(b/n)$.

Condition 2) is easy to show because we can always add any number of Gaussian random variables to the set $\{G_n(i,j), 1 \le i \le n\}$ with all the random variables having the desired first- and second-order moments. By the definition of exchangeability this new augmented set has members that are still exchangeable.

For 1) we want to find the values of m where

$$\lim_{n\to\infty} n^2\alpha_2(n) - b^2 = 0. \tag{A.8}$$

Let $f_\rho(x,y)$ be the bivariate Gaussian density function with marginals having mean 0 and variance 1, and correlation ρ. Also let $F_\rho(Z)$ be the distribution function of $Z = \max(X,Y)$ where X and Y are the marginals of the bivariate Gaussian distribution with correlation ρ. Let $\gamma = 1/(n-1)$. Then (A.8) is equivalent to

$$\lim_{n\to\infty} n^2 F_\gamma(x_n) = b^2. \tag{A.9}$$

Note that

$$F_\gamma(x_n) = \int_{-x_n}^{\infty} dy \int_{-x_n}^{\infty} f_\gamma(x,y)\,dx. \tag{A.10}$$

We first look at the inner integrand. $f_\gamma(x,y)$ can be expanded as a product of the two marginal density functions ($f(x)$ and $f(y)$), and a series expansion involving Hermite polynomials $H_i(x)$ [37]

$$f_\gamma(x,y) = f(x)f(y)\sum_{i=0}^{\infty}\frac{\gamma^i}{i!}H_i(x)H_i(y) \tag{A.11}$$

where

$$H_i(x) = (-1)^i e^{x^2/2}\frac{d^i}{dx^i}[e^{-x^2/2}].$$

Equation (A.10) can then be written as

$$F_\gamma(x_n) = \int_{-x_n}^{\infty} dy \int_{-x_n}^{\infty} f(x)f(y)\sum_{i=0}^{\infty}\frac{\gamma^i}{i!}H_i(x)H_i(y)\,dx \tag{A.12}$$

$$= \sum_{i=0}^{\infty}\frac{\gamma^{i+1}}{(i+1)!}[f(-x_n)H_i(-x_n)]^2 + Q(-x_n)^2. \tag{A.13}$$

Note that $Q(-x_n) = b/n$ and, by setting $x_n = -\sqrt{a\log n}$, we have that

$$f(-x_n) = \frac{1}{\sqrt{2\pi}}n^{-a/2}. \tag{A.14}$$

From (A.14) as $n \to \infty$

$$n^2F_\gamma(x_n) - b^2 = \frac{1}{2\pi}n^{1-a} + \frac{a}{4\pi}n^{-a}\log n + O\left(\frac{1}{n}\right). \tag{A.15}$$

Equation (A.9) is satisfied when $a > 1$. If we set $m = N/c\log N$, then $x_n = -\sqrt{c\log n}$. If $c > 1$ then for all $n \ge N$, it is easily shown that $x_n = -\sqrt{c_n\log n}$ where $c_n > 1$. Therefore, (A.15) is satisfied for all $n \ge N$ provided that $m < N/\log N$. Recall that b and therefore N are independent of the covergence in distribution of K_n to a Poisson random variable with parameter b. Using these facts, we can therefore state that for $m < N/\log N$ the number of events $\{G_N(i,j) < x_N\}$, $1 \le i \le N$ occurring converges in distribution (as $N \to \infty$) to a Poisson distribution with parameter $NQ(-x_N)$.

To use Theorem A2 to evaluate

$$\Pr(\min(g(i,j), 1 \le i \le N, 1 \le j \le m) < -N+1) = \Pr(\min(G_N(i,j), 1 \le i \le N, 1 \le j \le m) < x_N),$$

we again need two conditions analogous to those just obtained for all $n \ge N$. For Gaussian exchangeable random variables the second condition is trivial to show, just as in the previous case. For the first condition we require that

$$\lim_{n\to\infty} n^2m^2(\alpha_2(n)) = b^2. \tag{A.16}$$

For this case $x_n = -Q^{-1}(b/nm)$ and $b = NmQ(-x_N)$. We again use the Hermite polynomial expansion of the bivariate density to show that

$$\alpha_2(n) - Q^2(x_n) = \frac{1}{mn-1}\sum_{i=0}^{\infty}\frac{(n-1)\gamma_1^{i+1} + (m-1)\gamma_2^{i+1} + (nm-n-m+1)\gamma_3^{i+1}}{(i+1)!}[f(-x_n)H_i(-x_n)]^2 \tag{A.17}$$

where $\gamma_1 = 1/(n-1)$, $\gamma_2 = 1/(m-1)$, and $\gamma_3 = 1/(n-1)(m-1)$. From (A.14) and (A.16) as $n \to \infty$

$$n^2m^2(\alpha_2(-n) - Q(\beta)) = \frac{3}{2\pi}\frac{n^{2-a}}{a\log n} + \frac{a\log n}{4\pi}\left[n^{1-a} + \frac{n^{1-a}}{a\log n} + n^{-a}\right] + O(n^{-a}(a\log n)^3). \tag{A.18}$$

Equation (A.9) is satisfied when $a \ge 2$. Using the same arguments as in the first case, we can state that for $m \le N/2\log N$ the number of events $\{G_N(i,j) < x_N\}$, $1 \le i \le N$, $1 \le j \le m$ occurring converges in distribution (as $N \to \infty$) to a Poisson distribution with parameter $NmQ(-x_N)$. We have thus proved the following theorem which is identical to the theorem in Section III.

Theorem A3: 1) If

$$m < \frac{N}{\log N} \qquad X_N = \sum_{j=1}^{N} I(g(j,k) \le -N)$$

and Y_N is a Poisson random variable with parameter $\lambda_1 = NQ(\sqrt{N/m})$, then $X_N \to Y_N$ in distribution.

2) If

$$m \le \frac{N}{2\log N} \qquad X_N = \sum_{j=1}^{N} \sum_{k=1}^{m} I(g(j,k) \le -N)$$

and Y_N is a Poisson random variable with parameter $\lambda_2 = Nm\bar{Q}(\sqrt{N/m})$, then $X_N \to Y_N$ in distribution.

Appendix II

Given the random variables $\{u(j,k),\ 1 \le j \le N\}$ defined in Section III, we want to find the following probability:

$$\bar{p} = \Pr\left(\bigcap_{j=1}^{N} u(j,k) \ge -N+1 \right), \qquad 1 \le k \le m \quad \text{(B.1)}$$

where $\{u(j,k)\}$ are binomial random variables with $E(u(j,k)) = 0$ and

$$E(u(j,k)u(l,k)) = \begin{cases} (N-1)(m-1), & j = l \\ m-1, & j \ne l. \end{cases}$$

From (3.2) we note that each $u(j,k)$ is the sum of $m-1$ independent identically distributed (i.i.d.) random variables. Let us normalize these random variables, defining

$$U(j) = \frac{u(j,k)}{\sqrt{E(u(j,k))^2}}, \qquad 1 \le j \le N.$$

For large m we know that $U(j)$ converges to a standard Gaussian random variable (mean 0 and variance 1) by the central limit theorem. This theorem is also applicable to multivariate distributions; the N-variate joint distribution of $\{U(j),\ 1 \le j \le N\}$ thus converges to the multivariate Gaussian distribution $N(0, \boldsymbol{R})$ where

$$\boldsymbol{R}(i,j) = \begin{cases} 1, & i = j \\ \dfrac{1}{N-1}, & i \ne j \end{cases}$$

as m grows large; see [29, 38, 39].

Under the assumption that the $\{U(j)\}$ are Gaussian random variables, the results of Appendix I show that

$$e^{-NQ(\beta)} \approx \bar{p}_G = \int_{-\beta}^{\infty} \cdots \int_{-\beta}^{\infty} \int_{-\beta}^{\infty} (2\pi|R|)^{N/2} \cdot e^{-(1/2)x^T R^{-1} x}\, dx_1\, dx_2 \cdots dx_N \quad \text{(B.2)}$$

where $\beta = \sqrt{(N-1)/(m-1)}$ and $x^T = (x_1, \cdots, x_N)$. Since the Gaussian assumption is only true asymptotically, we are led to the question of whether or not $\bar{p} \to \bar{p}_G$. In this Appendix we show that $\bar{p} \to \bar{p}_G$ for large m by comparing the normal approximation of $\{U(j),\ 1 \le j \le N\}$ to $\{U(j),\ 1 \le j \le N\}$. We first state some necessary results from normal approximation theory.

A. Error Terms in the Normal Approximation to i.i.d. Random Vectors

Let us first look at the problem of approximating the distribution of sums of i.i.d. random variables by the normal distribution in R^1. We want to find how "close" the normal distribution is to the distribution of a normalized finite sum of i.i.d. random variables. Let $\{u_i,\ 1 \le i \le m\}$ be N i.i.d. random variables with $Eu_i = 0$, $Eu_i^2 = 1$, and let

$$U_m = \frac{1}{\sqrt{m}} \sum_{i=1}^{m} u_i.$$

Then the difference between $F_m(x)$, the distribution of U_m, and the standard normal distribution $\Phi(x)$ is given by

$$F_m(x) - \Phi(x) \approx \phi(x) \sum_{i=1}^{\infty} Q_i(x) m^{-i/2} \quad \text{(B.3)}$$

where $\phi(x)$ is the probability density function of the standard normal distribution and $Q_i(x)$ are polynomials derived from the standard Hermite polynomials [37]:

$$H_i(x) = (-1)^k e^{x^2/2} \frac{d^i}{dx^i} e^{-x^2/2}$$

with

$$Q_i(x) = \frac{(-1)^{i-1} c_i}{i!} H_{i-1}(x).$$

Letting $f_m(x)$ be the probability density function of $F_m(x)$, we have

$$c_i = (-1)^i \int H_i(x) f_m(x)\, dx.$$

We would like to find the order of magnitude of error terms in (B.3). In particular, we are interested in the size of $F_m(x) - \Phi(x) - [\phi(x)Q_1(x)/\sqrt{m}]$.

For lattice (i.e., discrete) distributions where all sample values can be expressed in the form $\alpha + ih$ where α and h are constants and i is an integer, the following result holds [40], [41]:

$$F_m(x) - \Phi(x) = \phi(x)\left(\frac{Q_1(x)}{\sqrt{m}} + \frac{hS_1(x\sqrt{m}/h)}{\sqrt{m}} \right) + o\left(\frac{1}{\sqrt{m}} \right) \quad \text{(B.4)}$$

where $S_1(\cdot)$ is a correction term arising from the discontinuities of the distribution function and h is the size of the lattice. The correction term is the periodic function

$$S_1(x) = x \bmod (1) - 0.5. \quad \text{(B.5)}$$

For distributions in R^N an approximation analogous to (B.4) is slightly more complicated. Now let $\{u_i\}$ be random vectors with $Eu_i = 0$ and $Eu_i^T u_i = \boldsymbol{R}$. Before stating a theorem from [38] we present some definitions. Let $f_m(t)$ be the characteristic function of the distribution $F_m(x)$. The characteristic function can be expressed in the following way:

$$f(t) = \exp\left(\sum_{|\nu|=1}^{\infty} \chi_\nu \frac{(it)^\nu}{\nu!} \right) \quad \text{(B.6)}$$

where ν is a nonnegative integer vector, $\nu! = \prod_{i=1}^{N} \nu_i!$, $(it)^\nu = \prod_{j=1}^{N} (it)^{\nu_j}$, and $|\nu| = \sum_{i=1}^{N} \nu_i$. The coefficients χ_ν are called the semi-invariants of F_m. For a Gaussian random vector all semi-invariants for $|\nu| > 2$ are 0. Also define

$$D^\nu f(x) = \frac{\partial f^{|\nu|}}{\partial x_1^{\nu_1} \partial x_2^{\nu_2} \cdots \partial x_N^{\nu_N}}(x) \quad \text{(B.7)}$$

and

$$D_j f(x) = \frac{\partial f}{\partial x_j}(x).$$

The following theorem from [38] can now be stated.

Theorem B1: Let $F_m(x)$ be defined as above. Then

$$\sup_{x \in R^N} |F_m(x) - \Phi_{0,R}(x)| = m^{-1/2} \sup_{x \in R^N} \left| \sum_{j=1}^{N} h_j S_1\left(x_j \sqrt{m}/h_j\right) D_j \Phi_{0,R}(x) + \sum_{|\nu|=3} \frac{\chi_\nu}{\nu!} D^\nu \Phi_{0,R}(x) \right| + o(m^{-1/2}). \quad \text{(B.8)}$$

Equation (B.8) is similar to (B.4) in that the first term on the right side of the equation is an error term due to the lattice distribution, and the second term describes the standard $O(m^{-1/2})$ error that occurs for all distributions.

B. Using Normal Approximation Theory to Show that $\bar{p} \to \bar{p}_G$

We now show that $\bar{p} \to \bar{p}_G$. For $U(j)$ we have $h = h_j = 1/\sqrt{N-1}$. We define the νth moment as

$$\mu_\nu = \iint \cdots \int x^\nu f_m(x)\, dx_1\, dx_2 \cdots dx_N \quad \text{(B.9)}$$

where $x^\nu = \prod_{i=1}^{N} x_i^{\nu_i}$ and $f_m(x)$ is the N-variate joint density function of $\{U(j),\ 1 \le j \le N\}$. For the case where $\chi_\nu = 0$ for $|\nu| = 1$ it is easily shown that $\mu_\nu = \chi_\nu$ when $|\nu| = 3$. The semi-invariants we need to use in (B.8) are taken from the following third moment values:

$$EU(i)U(j)U(k) = \begin{cases} 0, & i = j = k \\ \dfrac{2(N-2)}{\sqrt{(N-1)^3(m-1)}}, & i = j \ne k \\ \dfrac{2}{\sqrt{(N-1)^3(m-1)}}, & i \ne j,\ i \ne k,\ j \ne k. \end{cases} \quad \text{(B.10)}$$

We also observe that

$$D_j \Phi(x) = \int_{-\infty}^{x_1} \cdots \int_{-\infty}^{x_{j-1}} \int_{-\infty}^{x_{j+1}} \cdots \int_{-\infty}^{x_N} \phi(u_1, \cdots, u_{j-1}, x_j, u_{j+1}, \cdots, u_N)\, du_1 \cdots du_{j-1}\, du_{j+1} \cdots du_N \le \frac{1}{\sqrt{2\pi}} e^{-x_j^2/2} \quad \text{(B.11)}$$

where $\phi(x_{i_1}, \cdots, x_{i_k})$ is the joint Gaussian density function associated with random variables $\{U_j,\ 1 \le j \le k\}$. Then if we let $\nu_1^T = \{1,1,1,0,\cdots,0\}$ it is easily shown that

$$D^{\nu_1} \Phi(x) \le \phi(x_1, x_2, x_3). \quad \text{(B.12)}$$

Letting $\nu_2^T = \{2,1,0,\cdots,0\}$,

$$|D^{\nu_2} \Phi(x)| \le \frac{(1+(N-2)\rho)x_1 - 2x_2\rho}{(1+(N-1)\rho)(1-\rho)} \phi(x_1, x_2) + \frac{(1+2\rho)\rho}{(1+(N-1)\rho)(1+\rho)} \sum_{i=3}^{N} \phi(x_1, x_2, x_i) \quad \text{(B.13)}$$

where $\rho = E(x_1 x_2) = 1/(N-1)$. For our problem $x_j = \beta$ and $m = N/a \log N$. From (B.8) we have that

$$|\bar{p} - \bar{p}_G| = |F_m(\beta) - \Phi(\beta)| \le m^{-1/2} \left| \sum_{j=1}^{N} h_j S_1\left(\beta \sqrt{m}/h_j\right) D_j\left(\Phi_{0,R}(\beta)\right) \right| + m^{-1/2} \sum_{|\nu|=3} \left| \frac{\chi_\nu}{\nu!} D^\nu \Phi_{0,R}(\beta) \right| + o(m^{-1/2}). \quad \text{(B.14)}$$

Using the fact that the random variables we are looking at are exchangeable and using (B.10)–(B.13), we have that

$$|\bar{p} - \bar{p}_G| \le \sqrt{\frac{N}{2(m-1)\pi}}\, e^{-\beta^2/2} + \frac{N(N-1)(N-2)}{\sqrt{(N-1)^3(m-1)^2}} \cdot \left\{ \frac{(1+(N-2)\rho)\beta - 2\beta\rho}{(1+(N-1)\rho)(1-\rho)} \phi(\beta,\beta) + \frac{(1+2\rho)\rho}{(1+(N-1)\rho)(1+\rho)} (N-2)\phi(\beta,\beta,\beta) + \frac{1}{3}\phi(\beta,\beta,\beta) \right\} + o(m^{-1/2}). \quad \text{(B.15)}$$

Substituting values of m and β and simplifying, we have

$$|\bar{p} - \bar{p}_G| \le \sqrt{\frac{a \log N}{2\pi}}\, N^{-a/2} + N^{(1/2)-a} \frac{(a \log N)^{3/2}}{2\pi} + \frac{N^{(1/2)-(3a/2)} 5a \log N}{6(2\pi)^{3/2}}. \quad \text{(B.16)}$$

When $1/2 < a \ll N/\log N$, $|\bar{p} - \bar{p}_G| \to 0$ as $N \to \infty$. Asymptotically, we are only interested in the area around $a = 2$. For $a < 2$, $\bar{p}_G \to 1$ and for $a \ge 2$, $\bar{p}_G \to 0$. Therefore, we can state that

$$\bar{p} \approx e^{-NQ(\beta)} \quad \text{(B.17)}$$

as N grows large.

REFERENCES

[1] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas imminent in nervous activity," *Bull. Math. Biophys.*, vol. 5, pp. 115–133, 1943.

[2] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Nat. Acad. Sci. USA*, vol. 79, pp. 2554–2558, 1982.

[3] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing: Exploration is in the Microstructure of Cognition*, vols. 1 and 2. Cambridge, MA: MIT Press, 1986.

[4] S. Grossberg, "How does the brain build a cognitive code?" *Psychol. Rev.*, vol. 87, pp. 1–51, 1980.

[5] J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problems," *Biol. Cybern.*, vol. 52, pp. 141–152, 1985.

[6] D. W. Tank and J. J. Hopfield, "Simple 'neural' optimization networks: an A/D converter, signal decision circuit, and a linear programming circuit," *IEEE Trans. Circuits Syst.*, vol. CAS-33, pp. 533–541, 1986.

[7] M. Sivilotti, M. Emerling, and C. Mead, "A VLSI implementation of large Hopfield style neural networks," in *AIP Conf. Proc. 151, Neural Networks for Computing*, Snowbird, UT, 1986, pp. 408–414.

[8] H. P. Graf *et al.*, "VLSI implementation of a neural network memory with several hundred neurons," in *AIP Conf. Proc. 151, Neural Networks for Computing*, Snowbird, UT, 1986, pp. 182–188.
[9] D. Psaltis and Y. S. Abu-Mostafa, "Computation power of parallelism in optical computers," Dep. Elec. Eng., Calif. Inst. Technol., Pasadena, preprint, 1985.
[10] D. Psaltis, J. Hong, and S. S. Venkatesh, "Shift invariance in optical associative memories," Dep. Elec. Eng., Calif. Inst. Technol., Pasadena, preprint, 1986.
[11] Y. S. Abu-Mostafa and J. M. St. Jacques, "Information capacity of the Hopfield model," *IEEE Trans. Inform. Theory*, vol. IT-31, pp. 461–464, 1985.
[12] R. J. McEliece, E. C. Posner, E. R. Rodemich, and S. S. Venkatesh, "The capacity of the Hopfield associative memory," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 461–482, 1987.
[13] D. O. Hebb, *The Organization of Behavior*. New York: Wiley, 1949.
[14] T. Kohonen, *Self-Organization and Associative Memory*. Berlin, Germany: Springer-Verlag, 1984.
[15] ____, *Content Addressable Memories*. New York: Springer-Verlag, 1980.
[16] W. A. Little, "The existence of persistent states in the brain," *Math. Biosci.*, vol. 19, pp. 101–120, 1974.
[17] W. A. Little and G. L. Shaw, "Analytic study of the memory storage capacity of a neural network," *Math. Biosci.*, vol. 39, pp. 281–290, 1978.
[18] S. Amari, "Neural theory of association and concept formation," *Biol. Cybern.*, vol. 26, pp. 175–185, 1977.
[19] ____, "A method of statistical neurodynamics," *Kybernetik*, vol. 12, pp. 201–215, 1974.
[20] K. Nakano, "Associatron: A model of associative memory," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-2, pp. 380–388, 1972.
[21] G. E. Hinton and J. A. Anderson, *Parallel Models of Associative Memory*. Hillsdale, NJ: Erlbaum, 1981.
[22] D. J. Willshaw, O. P. Buneman, and H. C. Longuet-Higgins, "Non-holographic associative memory," *Nature*, vol. 222, pp. 960–962, 1969.
[23] R. Hecht-Nielson, "Artificial neural system technology," TRW preprint, 1986.
[24] S. S. Venkatesh and D. Psaltis, "Information storage and retrieval in two associative nets," Dep. Elec. Eng., Calif. Inst. Technol., Pasadena, preprint, 1985.
[25] S. S. Venkatesh, "Epsilon capacity of neural networks," Dep. Elec. Eng., California Institute of Technology, Pasadena, preprint, 1986.
[26] N. Rochester, J. H. Holland, L. H. Haibt, and W. L. Duda, "Tests on a cell assembly of the action of the brain, using a large digital computer," *IRE Trans. Inform. Theory*, vol. IT-2, no. 3, pp. 80–93, 1956.
[27] A. Kuh, "Stochastic models for interacting systems," Ph.D. dissertation, Princeton Univ., Princeton, NJ, 1986.
[28] W. Feller, *An Introduction to Probability Theory and its Applications*, vol. 1. New York: Wiley, 1950.
[29] ____, *An Introduction to Probability Theory and its Applications*, vol. 2. New York: Wiley, 1971.
[30] D. G. Kendall, "On finite and infinite sequences of exchangeable events," *Studia Sci. Math. Hungar.*, vol. 2, pp. 319–327, 1967.
[31] C. J. Ridler Rowe, "On two problems of exchangeable events," *Studia Sci. Math. Hungar.*, vol. 2, pp. 415–418, 1967.
[32] J. Galambos, "A general Poisson limit theorem of probability theory," *Duke Math. J.*, vol. 40, pp. 581–586, 1973.
[33] Y. S. Chow and H. Teicher, *Probability Theory: Independence, Interchangeability, Martingales*. New York: Springer-Verlag, 1978.
[34] B. De Finetti, "Funzione caratteristica di un fenomeno aleatorio," *Atti. Accad. Naz. Lincei. Rend. Sl. Sci. Fis. Mat. Nat.*, (6) 4, pp. 86–133, 1930.
[35] E. Hewitt and L. J. Savage, "Symmetric measures on Cartesian products," *Trans. Amer. Math. Soc.*, vol. 80, pp. 470–501, 1955.
[36] A. Renyi and P. Revesz, "A study of sequences of equivalent events as special stable sequences," *Publ. Math. Debrecen.*, vol. 10, pp. 319–325, 1963.
[37] J. B. Thomas, *An Introduction to Statistical Communication Theory*. New York: Wiley, 1968.
[38] R. N. Bhattacharya and R. R. Rao, *Normal Approximation and Asymptotic Expansions*. New York: Wiley, 1976.
[39] H. Cramer, *Random Variables and Probability Distributions*. Cambridge Univ. Press, 1937.
[40] C. G. Esseen, "Fourier analysis of distribution functions. A mathematical study of the Laplace–Gaussian law," *Acta Math.*, vol. 77, pp. 1–125, 1945.
[41] A. C. Berry, "The accuracy of the Gaussian approximation to the sum of independent variates," *Trans. Amer. Math. Soc.*, vol. 48, pp. 122–136, 1941.
[42] H. O. Lancaster, "The structure of bivariate distributions," *Ann. Math. Statist.*, vol. 29, pp. 719–736, 1958.

Stochastic Models for Interacting Systems

Anthony Kuh

ABSTRACT

This dissertation is concerned with the behavior of various systems of interacting components. Interacting systems have been used as models in Economics, Biology, Physics, Computer Science, and Engineering. We focus on discrete time systems, where components take on values from a common state space (usually a binary state space). Components, or cells, are updated at discrete time instants, with the updates on a cell depending on the previous value of some specified set of cells. The components of these interacting systems can be viewed as simple processing units, and the whole system can be viewed as a parallel computing machine. One of the main goals of the thesis is to find measures for the computational capabilities of different types of interacting systems.

We first study stochastic models for local spatially interacting systems (LSIS). A candidate mathematical model for local interacting systems is the class of Markov Random Fields (MRF). After discussing existing MRF models for LSIS, we introduce a discrete time synchronous model called the Completely Causal Markov Model (CCMM). Techniques are developed to analyze the behavior of the CCMM

and assess the computational capabilities of various models.

The class of Hopfield Associative Memory Networks (AMN) is discussed. Like LSIS, AMN consist of a number of simple interacting components, but unlike LSIS, AMN components usually have a high degree of interconnectivity. AMN models have been used to model neural networks. They have powerful computational capabilities, and they have been used to solve complex combinatorial optimization problems. We focus on assessing the computational capabilities of AMN models. The asymptotic information storage capacity of a simple AMN model is derived, using results from exchangeability and normal approximation theory. Other models for AMN are also developed along with an evaluation of tne computational capabilities of these networks.

Finally, we present an application for some models of LSIS, dealing with detecting faults that occur on semiconductor memory chips. LSIS are used to model Pattern Sensitive Faults (PSF) which occur when a read or write operation is faulty for some particular storage location when the memory cells exhibit a certain pattern.

Topics in Neural Networks

Princeton University

June 1988

Abstract

This dissertation considers some aspects of the behavior of several neural networks. We examine the original Hopfield Associative Memory (HAM) and derive a lower bound on the number of spurious minima when the stored memories are orthogonal. Two locally interconnected variations of the basic HAM network are proposed in which the maximum distance between two neurons that can be connected is upper-bounded by B. We show that for such locally interconnected networks containing N neurons, if $B/N \rightarrow 0$ as $N \rightarrow \infty$ then the capacity of the network is determined by B and is independent of N.

A macroscopic analysis technique first proposed by Amari for networks with random, nonsymmetric connection weights is modified to show that HAMs must have either one or two macroscopic stable states. The original analysis is also extended to networks of McCulloch-Pitts neurons with symmetric connections. The analysis and simulations show that the macroscopic behavior of networks with symmetric and nonsymmetric connections are qualitatively similar: the network either has one stable equilibrium; has two stable equilibria; or oscillates between two states.

We propose a new class of neural networks which are derived from the trellis graph representation of a convolutional code. Such a trellis network can be viewed as a collection of winner-take-all networks that are interconnected to reflect the structure of the trellis graph. We demonstrate by simulations that a trellis network with suitably defined connection weights can decode convolutional coded signals with added errors for a range of low error rates. We show that each of the subnetworks is stable for all choices of parameters. For a restricted set of parameters and a monotonically increasing gain with

sufficiently large derivative, we show that the entire network is equiasymptotically stable.

We propose a modified form of the trellis network which can tolerate errors in the input and replace failed neurons. Spare neurons are added to each of the subnetworks so that if a neuron fails, it can be replaced by any spare neuron in the same subnetwork. Replacement occurs without any supervision through modification of the connections between the subnetworks and has been verified by simulations.

Thomas Petsche
Siemens Corporate Research and Support, Inc.
105 College Road East
Princeton, NJ 08540-6668
609-734-3392
(petsche@siemens.com)